Trabajo de Fin de Grado

Valoración de startups con Aprendizaje Automático

Tutor de empresa:
Pablo de Manuel Triantafilo

Tutor de ADE:
Carlos Bellón Nuñez-Mera

Tutor de Informática:
Fernando Fernández Rebollo

Autor:
Víctor García Cazorla

Junio, 2016

Agradecimientos

A mi madre, padre y hermano, por apoyarme y estar a mi lado.

A mi empleador Pablo de Manuel Triantafilo, por darme la oportunidad de trabajar a su lado e introducirme de una manera fantástica al mundo laboral.

A mi tutor Carlos Bellón Nuñez-Mera, por ser el profesor más apasionado por lo que hace que he tenido.

A mi tutor Fernando Fernández Rebollo, por sus prácticos consejos.

A Jorge Morate Vázquez, por apoyarme y compartir tantos buenos momentos conmigo.

A Alejandro Antón Aguilar, por ayudarme y compartir conmigo su pasión por el Aprendizaje Automático y las Finanzas Corporativas.

Al resto de mis compañeros de universidad: Eduardo Díaz Gaditano, Diego Martínez Gálvez y Álvaro Montero Casarejos, por haber hecho de la universidad una bonita etapa de mi vida.

A Jacobo Calvo y el resto de la plantilla de Global Incubator por compartir conmigo una cultura en la que trabajar también puede ser divertido.

Abstract

NOTE: This report is primarily written in spanish. However, due to the fact that the author is required to prove his proficiency in english, this abstract, chapter 1 and chapter 9 will be written in this language.

In this project the author will develop a large-scale automatic valuation method for companies (with a strong focus on startups). The main challenge is the lack of financial information which is usually used when valuating companies with traditional methods. This problem will be overcome using a Machine Learning approach. The models will be fitted using collected data from sources from the Internet using web scrappers & crawlers. After a holistic analysis on the theoretical valuation of companies, it will be concluded that the valuation formula will lay its basis on the Discounted Free Cash Flows model. Also, a research of state-of-the-art Machine Learning techniques will take place. Then, an estimation model for each one of the formula's inputs will be built accordingly with the company's information that is known beforehand: in order to determine the company's economic activity from its description an expert rule-based system will be implemented. The company's revenues will be estimated using a meta-estimator that fits 700 randomized decision trees with data related to the number of employees, economic activity, social traction and other features. A similar estimator will also be used, with the help of an expert system, to transform those revenues into free cash flows. On the other hand, the risk of failure, growth and discount rate will be determined with the development of a system that crawls people's interest trends and matches the already known metrics of companies with the target ones according to its economic activity similarity.

Keywords

startup valuation, machine learning, artificial intelligence, discounted cash flows, corporate finance, supervised learning, ensemble-methods, extratrees, web scrapping, python, crisp-dm

Índice general

Índice de figuras

Índice de cuadros

Listado de Códigos

Listado de Pseudocódigos

Parte I

Introduction

Capítulo 1

Presentation, motivation and objectives

This is the report of the final university project carried out by the author as his core task at Global Incubator since the internship started in July 2015. The project puts the end to the Double Degree in Computer Science and Engineering and Business Administration with minor in Computer Science that the author started in 2011.

The author had never worked in the industry before, so this was a great opportunity to put together all the knowledge acquired during the last 5 years in a practical manner. The main fields related with this work are Corporate Finance and Machine Learning (both subjects were coursed in the second semester of the fifth grade). However, the key assets that have made the development of this project possible have been the ability to pursue and persist in learning, to organise self-learning, including through effective management of time and information, both individually and in groups. As a matter of fact, 'learn to learn' probably is the most valuable competence acquired at university.

1.1. Presentation

1.1.1. Capital Certainy

Global Incubator, among eight other startup[1] initiatives, belongs to the portfolio of Capital Certainty[2], a Seed Capital group whose mission is to build a world where eradicating poverty is profitable for companies. The group's rule of thumb is to identify, strengthen and develop initiatives that doesn't only offer economic returns but also have positive **social impact**. In order to materialize those social objectives into sustainable business models, Capital Certainty's workers usually perform thorough analyses and innovative designs at the cutting-edge of the most disruptives technologies. As a matter of fact, Capital Certainty is commited with Artificial Intelligence and its role in today's digital revolution.

[1] Entrepreneurial venture that *usually* is a fast-growing business and *sometimes* develops a scalable business model.

[2] Capital Certainty, S.L. overview: `http://portal.uc3m.es/portal/page/portal/investigacion/parque_cientifico/empresas/vivero/directorio_empresas/CapitalCertainty`

Figura 1.1: *Parque científico Leganés Tecnológico*, *Universidad Carlos III de Madrid* and the Capital Certainty group

The group, despite of currently having part of its headquarters in Silicon Valley, now coordinates its functions and operations from *Parque científico UC3M*[3] in Leganés, Madrid, an incubator[4] provided by *Universidad Carlos III de Madrid* that enhances knowledge and technological transfer from the university and promotes innovative entrepreneurial ventures.

1.1.2. Global Incubator

For its part, Global Incubator was founded in March 2010 with the objective of helping institutions to design, virtualize, manage and accelerate their innovation ecosystems. As a cloud computing software company, its main product is white-labeled software that perfectly fits the SaaS (Software as a Service) characteristics [13c]:

- Web access to commercial software
- Software is managed from a central location
- Software delivered in a "one to many" model
- Users not required to handle software upgrades and patches
- Application Programming Interfaces (APIs) allow for integration between different pieces of software

[3]Parque científico UC3M: `http://www.uc3m.es/ss/Satellite/UC3MInstitucional/es/PortadaMiniSiteA/1371207248804/Parque_cientifico`

[4]Oranization designed to accelerate the growth and support venture entrepreneurials success by providing a set of diverse resources, such as office space, cleaning services, networks etc.

In a not too distant future the company will implement a Platform As a Service model (PaaS). For now, the applications that are sold to the clients run on a cloud platform which sits on top of a server-based infrastructure. Many of the coworkers at Global Incubator have plenty of expertise in the full-stack development of software applications. Therefore, they take good care in maintaining and improving the whole stack of the company's cloud based technology that was once built from scratch by themselves.

Needless to say that the office's pleasant atmosphere at Global Incubator has been successfully proven to be a tremendous source of knowledge from which to learn, specially for a last-year student. Moreover, it has given the author a real first-hand insight into the software industry, enhancing the intern not only to learn state-of-the-art techniques and methodologies but to pull together and condense the knowledge gained from college.

1.2. Motivation

1.2.1. Entrepreneurial boom

During the last decade our world has experimented huge improvements in technology. Many quotidian devices that are being used now just didn't exist twenty years ago. As a matter of fact, the world is more connected than ever thus enhancing humanity's knowledge to spread over and flow trough it at high rates. Whenever some discovery takes place, no matter where it does, it propagates almost instantly directly to whomever is interested. Both physical and virtual nets built by humans gets more efficient year by year and the outcome of this process can be summarized in the following tendencies/facts:

- **Imagination is the limit**: People have more ideas due to the vast amount of knowledge that is easily accessible from almost any part of the world. Information flows more and better so citizens tend to be more aware of the current state of the world which makes it easier for them to come up with feasible ideas. Moreover, the technology development feeds back peoples ideas due to the fact that it makes more things possible to happen. So at the end the tendency is that the only limitation is people's own inspiration. The key idea here is these potential ideas arise when an entrepreneur realise that the once innovative ideas carried out by others can be integrated to produce a different outcome. In other words, a decentralized economy that permits and boosts people to act on their entrepreneurial insights will lead to an ecosystem where more entrepreneurial insights will take place [Hol98].

- **Entrepreneurship as economic growth**: As such, starting up bussinesses can be understood as a trigger of economic growth [Sch08].

- **No frontiers**: A fully connected world means more potential markets. Internet does the hard part breaking through geographical limitations and business only have to care about adapting their products to a broader spectrum of cultures.

- **Cheaper and easier**: The amount of resources needed to start up a business idea have decreased substantially. This is specially relevant for software projects. Moreover, there exists lots of investors out there disposed to hear any promising ideas. In addition, companies now have a broad range of options in order to receive funding, from Venture Capital[5] firms to

[5]Type of private equity, a form of financing that is providad to emerging companies.

crowdfunding initiatives. This is specially significant for startups, which are usually related with the ICT industry and therefore have less developing costs associated that companies from other sectors. At the end this means that they find it easier to grow in the mid/long term.

- **Reinvented methodologies**: It doesn't make sense to spend various months elaborating a static business plan and then try to face today's dynamic world. Old school business plans have passed away through a gradual process of renovation aimed to a new paradigm, where a whole new set of tools and methodologies have born, such as Customer Development [13b], Bussines Model Generation [10] and Lean Startup (figure 1.2.1).

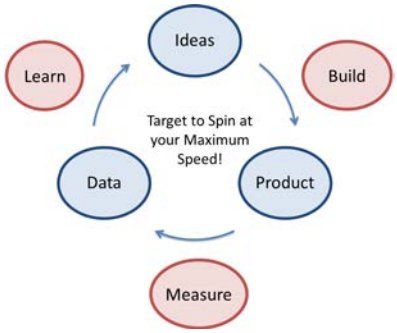


Figura 1.2: The lean startup methodology diagram [08a]

- **Collective power**: Starting up by creating a passionate team is a key-factor and easier than before. However, new trends and concepts like 'crowd-sourced labor' have appeared and suggest to turn the crowds and even the client into a collaborative workforce of an idea by successfully aligning the interests of each part.
- **Sensitive social media**: With all its pros and cons, the public likes to be aware of all that is going on out there. Social buying is a reality and people uses each others feedback constantly and thus its of big importance for a company to have enough social traction. The milestone is to build up a modern business that is friendly to the world.

This ingredients were sufficient to experiment an even bigger entrepreneurial boom than the dot-com bubble of 1997-2000. It is precise to consider, however, that this new wave of business activity is no older than two years, since it didn't really started to accelerate until the end of the Great Recession of 2008-2015. In order to realise the real growth of entrepreneurial activity of the recent years it is useful to know about the Kauffman Index. This indicator measures business creation in the United States, integrating several high-quality sources of timely entrepreneurship information into one composite indicator of startup activity that puts together primarily three components:

1. The rate of new entrepreneurs in the economy, calculated as the percentage of adults becoming entrepreneurs given a month.
2. The opportunity share of new entrepreneurs, calculated as the percentage of new entrepreneurs driven essentially by 'opportunity' vs 'necessity'.

3. The startup density of a region, measured as the number of new employer business normalized by population.

This startups activity index, as seen in the figure 1.2.1, rose in 2015, reversing a downward trend that started in 2010 in the middle of the Great Recession.

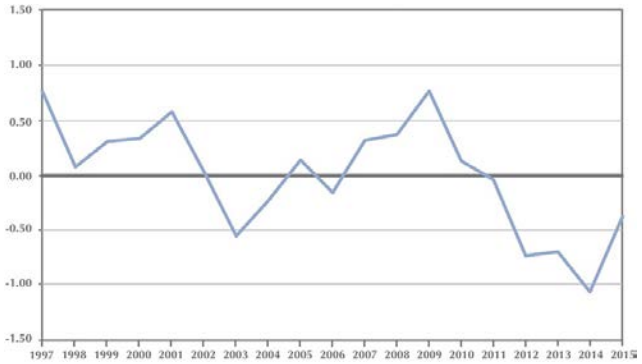


Figura 1.3: Kauffman Index: Startup Activity (1997-2015) [FMRR15]

1.2.2. The project

With this ecosystem full of companies trying to do its best to interact with each other and find investors it is obvious that there is a need of some order to benefit all the parts. There exists some startup platforms like *CrunchBase*[6], *Startup Genome*[7] or *AngelList*[8] that try to overcome this problem by providing startup communities. One one hand, companies sign up and use these platforms as a showcase so they get to be known by the world and increase their contacts in terms of investors, employees and even competitors. On the other hand, investors (specially Venture Capitals), accelerators and incubators also sign up voluntary in order to be up to date about any investing opportunities and also track deals and funding rounds of other firms.

The effective business model of these startup platforms/ecosystems lays its basis in a crowd-sourced labor strategy to acquire the data and usually spending also some resources in curating it before turning it into a salable product from which they earn profit just like any other private company.

However, there are some organisations that would like to buy this kind of product but need an extra layer that adds up some value. They not only want to get access to a bunch of startups and companies information but also being able to use some special features. These are the type of organizations that are Global Incubator's clients (related to this product) and these are some of their needs:

- Organise startup competitions/gatherings. This implies an intelligent process that selects the best and most disruptive startups that meet specific requirements, such as region, industry, social impact etc. that the client wants to define dynamically.

[6] `www.crunchbase.com`
[7] `www.startupgenome.co`
[8] `www.angel.co`

- Discover, stay up to date and get in touch with all the actors of the startup ecosystem in a friendly and customizable way, with notifications, alerts etc.
- Having an effective interface that allows them to perform strategic analysis, proofs of concepts and geo-innovation in a particular market of interest.
- Making smart and collaborative investments.
- Tracking worldwide trends and social traction of products and technologies.

Global Incubator had to develop a system able to fit its clients needs. The role of the author was to help fulfilling these needs. Specifically, these are the system features/tasks related with the startup project that the author have been involved with (among other projects) during his stay (full-time since December 2015) at Global Incubator:

- Competitor analyses (Crunchbase, Datafox, Unomy, Mattermark, Trendify, Angel.co, CB Insights...).
- Mockups and wireframes of 'startup details' view.
- Improvements of database in quality and quantity of data by the use of crawlers and web scrappers (more about this later).
- Implementation of a startup recommender system using Machine Learning. Both content-based [RRSK10] and collaborative-filtering [ERK10] types.
- Implementation of a company valuation method.
- Implementation of a trends observatory based on a set of web scrappers that crawl public interest growth related to any keyword according to different sources such as GitHub repositories, Wikipedia articles, Google Trends, articles written in TechCrunch, Twitter and funding rounds.

As shown, many of these tasks involve Data Analytics and Data Engineering skills that the author has developed while performing them at the company. Among all the activities implicated, this internship project will focus on the design of the valuation method. After putting aside the rest of the related work, this task will still fulfill/help many of the clients needs by its own, since it will at least provide a tool for ranking the companies, which is highly valuable for most of the clients of this product.

1.3. Objectives and overall strategy

Therefore, the main objective of this project is to create a valuation method for young companies, with strong focus on the use of Machine Learning and expert knowledge based tools. That is the principal aim of the work, and it can be divided into six specific objectives, which can be listed as the follows (1-4 are desired system & model capabilities and 5-6 are general purpose):

1. **Utility measure**: This is the main objective. First of all, as it has been said at the end of subsection 1.2.2, the principal and most precious feature is being able to rank companies (i.e. sorting any set of random businesses) according to some indicator that measures *some kind*

of value/utility. As it will be shown in following chapters, any valuation method is susceptible of opinion and there are many of them out there. So care must be taken if the one developed and designed here will be supported by any of them.

2. **Monetary**: The accomplishment of this goal implies the fulfillment of the first one. It is good for a valuation method to measure a company's value in monetary units. Besides of being able to rank companies, valuation matters to entrepreneurs because it determines the share of the company they have to give away to an investor in exchange for money. Note that specially when talking about small companies that may be years away from sales, the valuation is not supposed to actually reflect the real value of it but the potential that it has. Moreover, it is extremely hard to determine the accurate value of a company while it is in its infancy stages as its success or failure remains uncertain. Also, there may be other stakeholders in monetary valuation. For example, maybe a big enterprise that wants to expand its business operations by acquiring a startup.

3. **Scalable**: It will be also shown that the hardest gap to overcome is the fact that the amount of business-information required to implement any reasonable valuation method usually is highly company specific and should be analysed on a case by case basis. However, the system should not only be able to valuate *some* specific companies/startups. The results should be adhoc to each company but not the method nor the design of the inputs. It should be applicable on demand, for example when a new company pops up in the dataset and needs to be given a value, and also on massive batch mode. Global Incubator's startup dataset is 1.5M+ long and it is expected to keep growing.

4. **Integrated system**: the valuation method must become one of the business processes. This means that, besides of being composed by different modules, at the end it will have to behave like a unitary system with a clearly defined set of functionalities.

5. **Provides industry insights**: Capital Certainty, as a Capital Seed group is also interested in investing in promising ideas. It will be beneficial that this project could provide some insight of the business relations and patterns hidden in the companies dataset. For example, which correlations between the *value* of a company/startup and its attributes (location, industry, employees, social traction etc.) will be worth knowing.

6. **Learning path**: It is expected that the required work of this project will carry an interesting learning process of data Engineering and Machine Learning with it. Firstly, the author's professional career aims to the Data Science field. Secondly, his work as a novel Data Scientist at Global Incubator is involved with many other Data Analytics projects. Therefore, in order to amortize and maximize the gain of the learning curve it will be a relevant objective of this project to get fluent with the command of Data Mining tools, learn the know-how of state of the art concepts and acquire professional expertise in the field.

As it will be justified afterwards, the strategy that will be followed during the process of creation of the whole system will be:

1. Define the most suitable valuation method according to the circumstances. The chosen one will result from previously analysing which are the most used valuation methods in the industry and why. Each input of the formula must be clearly identified and understood.

2. Define the type of the problem and subproblems that are going to be addressed (which type of techniques are going to be used?). It turns out that many of the subproblems will be closely related with Data Analytics and Data Mining. Thus, a brief introduction and analysis of the state-of-the-art techniques for solving this kind of problems will also be exposed, with a precise focus on the specific algorithms that will be used.

3. Chose the appropriate methodology to follow according to the type of the problem. Due to its nature, this project will need a methodology with strong focus on experimentation phases and thus should include this basics steps: analysis, design, evaluation and deployment. Moreover, it will be useful if its iterative, in order to being able to reuse the feedback of a step to improve the work done in the previous ones. This presumably will lead to a better solution.

4. Solve each subproblem separately. The big problem should be firstly clearly divided into distinct pieces. Then, they will be tackled one after the other. The order used when solving them will be specially influential if more than one belong to the same pipeline. This includes all the steps of the chosen methodology, from early experiments to deployment, the full cycle. However, it must be noted that this project will have parts of different technique branches of Computer Science so each subproblem will be tackled from different adhoc perspectives. As it has been said, some of them may share characteristics and share the process methodology, but some may not.

Parte II

Teoría y Estado del Arte

Capítulo 2

Valoración de empresas

En finanzas, una valoración es el proceso de determinar el valor actual de un activo o una empresa. Existen distintas técnicas y no hay un acuerdo de cuál de ellas es la más adecuada en el 100 % de los casos.

Es un terreno con frecuentes subjetividades en el que muchas voces autorizadas han aportado su propia manera de percibir el valor de las cosas. Por ello, aunque sí que es cierto que existen algunos métodos que se utilizan más que otros en algunas situaciones concretas porque *se entiende* que son más certeros cuando el problema presenta ciertas características, no sería correcto afirmar que existe un 'Estado del Arte' como tal en lo que a valoración de empresas se refiere.

En este capítulo se explicarán brevemente cuáles son los fundamentos de la valoración de empresas con el objetivo de utilizar el conocimiento ya existente en la elaboración del método de valoración para este proyecto. Además, será también necesario estudiar cómo se adapta el método de valoración escogido al proyecto de Global Incubator.

2.1. Métodos relativos y absolutos

En lo que a valoración de empresas se refiere, existen principalmente dos familias de métodos: aquellos que estiman el valor relativo de las cosas mediante comparables y aquellos que estiman el valor absoluto mediante proyecciones hacia el futuro.

2.1.1. Valor relativo

Es el nombre que hace referencia al *valor* de un activo que un tipo concreto de técnicas calculan utilizando el valor previamente determinado de otros activos similares (normalmente por el mercado) [Damb].

El cálculo de valoraciones relativas es sencillo: primero se identifica un activo comparable a aquello que se quiere valorar y que ya ha sido valorado. Después, dado que no tendría sentido trabajar con valores absolutos, se estandariza o tipifica su valor obteniendo un múltiplo con el que sí se podrán hacer comparaciones (controlando en la medida de lo posible las diferencias existentes entre los 'activos comparables'). Un múltiplo mide un aspecto concreto de la salud financiera de una empresa y se determina dividiendo dos métricas de la propia compañía entre sí para luego

compararlo con el de empresas competidoras.

$$Múltiplo = \frac{Métrica \quad A}{Métrica \quad B}$$

Por ejemplo, el múltiplo posiblemente más usado en la valoración de *equity* es el *PE* (*Price to earnings ratio*) y se cálcula dividiendo el precio de mercado de la acción entre la porción del beneficio de la compañía asociada a ella. Por lo tanto, una compañía que tuviese un alto valor en el mercado[1] respecto a su nivel de ganancias tendrá un múltiplo mayor, y eso es un indicio de que la empresa puede estar sobrevalorada. Este múltiplo no contempla la emisión de deuda por la empresa. Si quisiese tener en cuenta, lo más común es utilizar el múltiplo *EBITDA*[2] que sale de dividir el valor de la empresa entre el *EBITDA*.

Una de las utilidades de este tipo de valoraciones es determinar si una empresa que cotiza en bolsa está sobre o infravalorada. La manera típica de hacer estos cálculos mediante el uso de múltiplos es sencilla:

1. Crear una lista de compañías comparables a la empresa objetivo de la valoración. A menudo se seleccionan aquellas empresas que sean del mismo sector/industria, operen en mercados similares y no sean demasiado distintas en número de empleados.

2. Obtener los valores de mercado actuales de las empresas listadas.

3. Calcular los múltiplos adecuados para cada una de ellas.

4. Comparar los resultados del paso anterior con los múltiplos calculados de la empresa objetivo y así poder deducir si está sobre o infravalorada.

Un aspecto importante de las valoraciones relativas es que es necesario establecer cuál es el múltiplo que tiene más sentido utilizar mediante un análisis del sector e industrias concretos a los que pertenece la compañía, por ejemplo, precio entre flujos de caja[3] para las inmobiliarias y precio entre ventas para empresas minoristas. El motivo es que cada uno de los múltiplos que se pueden calcular en un sector concreto tendrá unos valores medios potencialmente distintos a los de otro sector por lo que no sólo cada industria tiene ciertos múltiplos más estables que otros sino que a menudo carece de sentido utilizar múltiplos para comparar empresas de distintos sectores. En cualquier caso, siempre es buena práctica realizar un análisis estadístico de los múltiplos de un sector (calcular desviación típica, la mediana, como de relevantes son los datos atípicos o *outliers* etc.)

Por otro lado, a pesar de que su cálculo sea muy sencillo, el uso de estos comparables es inútil si el propio sector o industria están enteramente sobre o infravalorados. En esta misma línea, pueden ocurrir imprecisiones cuando una determinada empresa obtiene unos resultados atípicos haciendo que sus múltiplos se diferencien mucho de los del resto de su sector. En esta situación es difícil hacer una valoración relativa precisa y en muchas ocasiones es necesario esperar a que el propio mercado ajuste el valor de la empresa. Pero es muy difícil saber cuándo esto ocurre y frecuentemente los inversores realizan inversiones porque la empresa a priori parece que está infravalorada para

[1] Nivel de capitalización en el mercado de una empresa pública, equivalente al número de acciones emitidas multiplicado por el precio actual de cada una.

[2] Indicador financiero cuyas siglas significan *Earnings Before Interest, Taxes, Depreciation, and Amortization* (beneficio antes de intereses, impuestos, depreciaciones y amortizaciones).

[3] La cantidad neta resultante de las salidas y entradas de caja o efectivo en un negocio y periodo dados.

a continuación darse cuenta que la compañía de las acciones que han adquirido está apunto de colapsar. Todas las compañías, incluso aquellas pertenecientes a la misma industria, tienen unos patrones propios de crecimiento, riesgo y flujos de caja que determinan su múltiplo [Dam01] y por lo tanto sólo se deben aplicar los múltiplos cuando se tenga cierto nivel de seguridad en que dos compañías son realmente comparables.

Por todo ello, dado que en este proyecto se intentará valorar compañías de alta diversidad, se intentará no recurrir a los múltiplos. No obstante, como se explicará en el apartado 5.3 en ocasiones será necesario consultar algunos ratios medios a nivel sectorial para utilizarlos en las valoraciones.

2.1.2. Valor absoluto

En este tipo de valoraciones se proyectan los flujos de caja que generados para determinar el valor financiero de una determinada compañía.

Los inversores, además de comparar múltiplos, pueden estar interesados en calcular el valor total de una compañía especialmente cuando su motivación es adquirir una porción de las acciones de la empresa. Aparte de los flujos de caja, en este tipo de métodos es de especial importancia determinar la tasa de descuento hacia el presente[4] para efectivamente calcular el valor de los flujos proyectados al futuro.

La idea es calcular el valor actual de unos flujos de caja futuros descontados a una determinada tasa de descuento. Cuando esta tasa sea más alta, el valor presente de los flujos será menor. La idea de descontar flujos monetarios hacia el presente es muy común en las matemáticas financieras y se puede expresar con la fórmula 2.1:

$$DCF = \frac{E_0(CF_1)}{(1+r)^1} + \frac{E_0(CF_2)}{(1+r)^2} + ... + \frac{E_0(CF_\infty)}{(1+r)^\infty} = \sum_{n=1}^{\infty} \frac{E_0(CF_n)}{(1+r)^n} \tag{2.1}$$

donde $E_0(CF_n)$ son los flujos de caja del periodo n esperados en el periodo 0 (presente) y r la tasa de descuento aplicada.

Su principal desventaja es que las proyecciones no son 100 % precisas sino que cuanto más lejanas del momento presente más varianza se introduce en el modelo. No obstante, este método es muy utilizado a la hora de valorar empresas ya que supone una razonable estimación del valor intrínseco de un activo y además es válida para valorar empresas de cualquier sector. A pesar de que la fórmula en sí sea objetiva, las valoraciones que se hagan con ella también pueden ser buenas o malas y eso dependerá en gran medida de si la tasa de descuento r está bien ajustada o no. En cualquier caso, parece que éste método se ajusta más a los objetivos del proyecto listados en 1.3 por lo que sobre él se basarán las valoraciones efectuadas por el sistema.

2.2. Descuento de flujos libres de caja

A continuación se detallarán algunos aspectos relativos al método principal en el que se basarán las valoraciones: el descuento de flujos de caja.

En primer lugar en el término 'flujos de caja' hace referencia a todo el efectivo/caja que sale y entra de la compañía, pero para que la valoración se más precisa hay que sustraer algunos de sus componentes hasta que finalmente lo que se descuente sean los flujos libres de caja o FCF.

[4]Tasa de interés a alcanzar necesaria sobre una cantidad de dinero dada hoy para acabar teniendo otra cantidad de dinero (a menudo mayor) en el futuro. Está muy relacionada con el valor del dinero a lo largo del tiempo.

Existen varias maneras de expresar dichos flujos mediante una fórmula. En este proyecto se utiliza la siguiente ya que la manera en la que agrupa los factores resulta muy explicativa:

$$FCF = OCF - CAPEX$$

$$OCF = Revenues - OPEX$$

donde:

- **FCF**: El *Free Cash Flow* o flujo libre de caja es una medida de rendimiento financiero que representa el efectivo que una compañía es capaz de generar una vez que se ha desprendido del dinero requerido para mantener o expandir los activos de la misma. La presencia de flujos libres de caja positivos a menudo se relaciona con una buena salud financiera ya que indica que la empresa tiene efectivo para expandirse, desarrollar nuevos productos, recomprar acciones, pagar dividendos o reducir su deuda.

- **OCF**: El *Operative Cash Flow* o flujo de caja operativo es una medida de la cantidad de efectivo generado por las operaciones corrientes de una compañía. Es importante ya que indica si una compañía es capaz de generar suficientes flujos de caja positivos para mantener y aumentar sus operaciones o si por el contrario requiere financiación externa.

- **CAPEX**: *Capital Expenditure* son los fondos destinados por una compañía a adquirir o mejorar sus activos físicos con el habitual objetivo de expandir las capacidades de la empresa para generar beneficios. Así, por ejemplo caerían en esta categoría los gastos asociados a la compra de propiedades, ampliación de fábricas, reparación de equipos etc. No sólo eso, sino que también se incluyen en esta categoría los gastos en R&D (*Research and development*) y las adquisiciones de otras compañías. Por otro lado, es en este factor donde se imputa la depreciación de aquellos activos de la compañía cuya vida útil es superior a un año, acorde con la regulación impositiva de cada país. Lo correcto sería entonces hablar de ***Net** Capital Expenditure* o inversiones en bienes de capital netas, que representan la diferencia entre los gastos de capital y la depreciación. No obstante, para simplificar se mantendrá el término como *CAPEX*.

- **Revenues**: Los ingresos es la cantidad de dinero que una compañía recibe durante un determinado periodo, incluyendo los descuentos y deducciones por devoluciones de mercancías. Además, ya debe tener descontados los impuestos y los intereses.

- **OPEX**: *Operating Expenses* o gastos operativos son la categoría de gastos en los que una compañía incurre como resultado de sus procesos operativos de negocio habituales. A diferencia que el *CAPEX*, el *OPEX* es 100 % deducible en el mismo año en el que se produce. Por otro lado, es importante mencionar que se incluirán en éste término las inversiones en capital circulante anuales. El capital circulante (también conocido como fondo de maniobra) es una medida de la eficiencia de la compañía y su salud financiera a corto plazo. En términos contables se calcula como la diferencia entre el activo circulante (tesorería y cuentas financieras, cuentas a cobrar, existencias y otros activos a corto plazo) y el pasivo circulante (cuentas a pagar y provisiones a corto plazo).

Por otro lado, la tasa utilizada en el modelo clásico para descontar los flujos de caja es r_W, *WACC* o Coste Medio Ponderado del Capital. Como se desprende de su nombre, es una media

ponderada que tiene encuentra la estructura y las tasas de descuento de los tipos de capital de la empresa:

$$r_W = \frac{E}{V} \times r_E + \frac{D}{V} \times r_D \times (1 - T_c)$$

donde:

- **V**: El valor de mercado de la empresa.
- **E**: El valor de mercado de las acciones emitidas (*Equity*). Dentro pueden coexistir distintos tipos de acciones: ordinarias, preferentes etc.
- **D**: El valor de mercado de la deuda (*Debt*) que normalmente suele estar compuesta de bonos que la empresa ha emitido.
- $\boldsymbol{r_E}$: *Cost of equity* o coste de las acciones E, es el retorno que los accionistas le exigen a la compañía a cambio de haberse arriesgado al invertir en ella.
- $\boldsymbol{r_D}$: *Cost of debt* o coste de la deuda es la tasa a la cuál una compañía paga por haber emitido su deuda D.
- $\boldsymbol{T_c}$: Tasa de impuestos corporativos

Una manera de entender el WACC en términos prácticos es como una representación del coste de oportunidad en el que incurre el inversor al decidir arriesgarse a poner su dinero en una compañía. Tanto el WACC como los FCF son de naturaleza intrínseca lo que hace que el valor calculado también lo sea [Dam12]. Esto favorece que la valoración final se parezca a lo que un inversor realmente debería estar dispuesto a pagar por adquirir las acciones de la empresa, independientemente del valor de las demás compañías. Esta característica se alinea muy bien con los objetivos del proyecto.

Una vez analizados los principales componentes del modelo de descuento de los FCF al WACC es conveniente aclarar la manera en la que el riesgo de la empresa debería reflejarse en la fórmula. Como se explicará en las secciones 6.2 y 6.3 existen más de una manera de hacerlo y para ello será necesario descomponer el 'riesgo' en dos factores.

2.3. Adaptación de la fórmula al problema

Ya se tienen definidas las bases del método de valoración a utilizar pero todavía falta ajustarlo a las necesidades más específicas de este proyecto, y eso es justo lo que se va a hacer en esta sección. Como se adelanto en la sección 1.3, existen principalmente dos problemáticas a resolver: el hecho de la existencia de empresas jóvenes/*startups* en el dataset y la escasez de información financiera detallada para cada compañía.

Problemática 1: startups

Como se viene viendo en las secciones anteriores, la valoración de un negocio no es algo obvio en absoluto, para ninguna compañía. Este hecho se acrecenta en el caso de compañías que no son públicas, están en sus primeros pasos y no están generando ingresos. A continuación se lista con algo más de detalle los principales retos que implica valorar compañías tan jóvenes [Dam09]:

- Apenas existen históricos ni de su actividad operativa ni financiera.

- La mayoría o no tienen, o tienen muy pocos ingresos. A menudo incurren en pérdidas operativas.
- Existe mucho riesgo y la mayoría no sobreviven (como se verá posteriormente, esto depende de varias cosas).
- Hay muy pocas excepciones y casi todas las *startups* financian su *equity* mediante fuentes privadas y no públicas. Es común que en las primeras etapas las aportaciones de dinero provengan de *FFF*[5] pero según crecen empiezan a ser más atractivas para entidades de capital riesgo, quienes obtienen a cambio un porcentage de la empresa.

Estas son algunas de las maneras mediante las que ciertos inversores calculan valoraciones de *startups* en la actualidad:

- **Coste de duplicar**: Como sugiere el nombre, consiste en calcular el dinero necesario para replicar la compañía entera desde cero (incluyendo todos los activos). Tiene la ventaja de que el valor de la mayoría de los activos es una medida objetiva, pero no refleja el potencial que tiene la *startup* para crecer y generar retornos. No sólo eso, sino que además con este enfoque es realmente complicado reflejar activos intangibles como el capital intelectual o el valor de la marca.
- **Múltiplos**: a pesar de sus desventajas explicadas en la sección 2.1.1 y del hecho de que en el caso de las *startups* es todavía más difícil encontrar comparables, hay un número importante de grupos de inversión de Capital Riesgo que los usan.
- **Datos cualitativos**: En contra a todos los métodos descritos anteriormente, que son cuantitativos, algunos inversores prefieren utilizar información de un carácter más cualitativo. Por ejemplo: el tipo modelo de negocio, identificación del mercado objetivo, el plan de marketing, qué tipo de competidores existen etc. Existe por ejemplo una página web donde con carácter didáctico se puede hacer un test de 25 preguntas para acabar con una valoración monetaria [15a]. Aunque este tipo de información debería estar relacionada con el valor de una *startup*, por razones obvias no sería viable aplicarla masivamente de manera automática.
- **Flujos de caja**: Es también utilizada pero es difícil aplicarlo *de manera tradicional* ya que muchas *startups* todavía no generan ingresos. Normalmente el riesgo extra de estas compañías tan jóvenes se ve reflejado en la tasa de descuento.

En un tema tan importante como este no son pocas las veces que prestigiosos académicos en valoraciones corporativas han advertido del incorrecto uso de las herramientas financieras que hace el sector privado (especialmente la banca) y que a menudo resultan en valoraciones incorrectas [Fer07]. Advierten que, incluso en el caso de las *startups* -mucho más difíciles de valorar- los resultados que se obtienen son defectuosos y deberían ser reemplazados [Dam09].

Problemática 2: poca información financiera

En el dataset de compañías que se pretende valorar hay más de un millón de empresas, tanto *startups* como compañías maduras. Ahora bien, no de todas se dispone de la misma cantidad de información. Concretamente, hay muchas de ellas de las que no se tienen datos financieros por

[5] *Friends, family and fools.*

lo que habrá que estudiar de qué manera poder estimar aquella información que no se conoce de manera explícita. Como se explicará en posteriores capítulos, el Aprendizaje Automático jugará un papel clave aquí.

Fórmula final

El diseño final de la fórmula a aplicar es el siguiente

$$V = \frac{E_0(FCF)}{r_W - g} \tag{2.2}$$

Se puede apreciar que está basada en el modelo de descuento de descuento de flujos de caja al coste medio ponderado del capital WACC o r_W. No obstante, se ha utilizado una versión de la misma con forma de perpetuidad. Es decir, se ha decidido estimar los flujos de caja libres exclusivamente del primer periodo a la par que una tasa de crecimiento anual g de los mismos en lugar de proyectar un flujo concreto para cada periodo. Esta asunción supone que los flujos seguirán produciéndose hasta el infinito mientras crecen a una tasa constante g, de tal manera que $FCF_t = (1+g)FCF_{T-1}$.

Como se verá a lo largo de la parte de Construcción del Modelo de este trabajo, se contará con estimaciones de los siguientes inputs que serán combinados entre sí como sugiere la ecuación 2.4, que es la versión reducida de la ecuación 2.3:

$$V = \frac{SR \times (Revenue \times Ratio_1 - Revenue \times Ratio_2)}{r_W - g} \tag{2.3}$$

$$V = \frac{SR \times Revenue \times (Ratio_1 - Ratio_2)}{r_W - g} \tag{2.4}$$

donde:

- *Revenue* son los ingresos anuales de la empresa. Se estimarán con un regresor a partir de información relativa al tamaño de la empresa, el sector, la industria, palabras clave relacionadas con la actividad de la empresa, tracción social y algún dato más. Se explicará con más detalle en la sección 7.1.
- g es la tasa de crecimiento anual de los flujos libres de caja y se explicará en la sección 6.1.
- r_W es el WACC y como se explicará en la sección 6.2 se tomarán los múltiplos de cada industria.
- $Ratio_1 = \left(\frac{OCF}{Revenue}\right)$, es el ratio de los flujos de caja operativos respecto a los ingresos. Se estimará mediante un regresor con el objetivo de obtener los OCF y se explicará en la sección 7.2.
- $Ratio_2 = \left(\frac{CAPEX}{Revenue}\right)$, es el ratio de los gastos en inversiones de capitales sobre los ingresos anuales. Como se explicará em la sección 7.3, se tomarán los múltiplos de cada sector.
- SR es la probabilidad de éxito de la empresa o *Success Rate*. Como se explicará en la sección 6.3, irá en función principalmente de la edad de la compañía y de su actividad como empresa (sector, industrias etc.)

A continuación se hacen algunas aclaraciones acerca de la fórmula 2.4 y sus *inputs*:

- Todos las veces que se nombra a *Revenue* en los *inputs* se hace referencia a los mismos ingresos. No obstante, en el caso de $Ratio_1$ y $Ratio_2$ se estimará el ratio directamente, sin efectuarse la operación $\frac{OCF}{Revenue}$ ó $\frac{CAPEX}{Revenue}$ en ningún momento. Por eso no se ha seguido simplificado la fórmula 2.4.
- Aunque SR y de r_W aparezcan como *inputs* diferentes, su existencia no se puede explicar por separado. Concretamente, lo que se pretende es hacer una más precisa estimación del WACC teniendo en cuenta los distintos tipos de riesgo adecuadamente y ajustándose a los supuestos establecidos. Se explicarán en detalle conjuntamente en la sección 6.2.

Capítulo 3

Aprendizaje Automático

La realización de este proyecto no hubiese sido posible sin el Aprendizaje Automático. En este capítulo se pretende dar una visión de por qué el Aprendizaje Automático es relevante para la ciencia y la sociedad en general. A su vez, se espera poder transmitir al lector la intuición de por qué ha sido de tal importancia en este proyecto.

Se ofrecerá una visión general de qué es y por qué existe. Posteriormente se explicará la naturaleza de los problemas que se van a intentar resolver utilizando técnicas pertenecientes al estado del arte de este fascinante campo de la Inteligencia Artificial.

3.1. Qué es y por qué existe

3.1.1. Su papel en la sociedad

El Aprendizaje Automático o *Machine Learning* se encuentra mucho más presente en el día a día de las personas que lo que generalmente se piensa.

- Cada vez que una aplicación de correo electrónico detecta que un mensaje entrante es spam para mejorar la experiencia del usuario es porque detrás existe un algoritmo de Aprendizaje Automático capaz de tomar decisiones a partir del análisis de un texto.
- Cada vez que se busca algo en internet, el motivo por el que se obtienen más que razonables resultados en tan poco tiempo es porque los algoritmos que hay detrás de los principales motores de búsqueda han sido diseñados para aprender **automáticamente** a ordenar o establecer *rankings* de páginas web de manera eficaz.
- De la misma manera, cada vez que alguien sube una foto a una red social, la aplicación responsable es capaz de etiquetar automáticamente a las personas porque ha aprendido a localizar caras entre píxeles.

Conseguir unos resultados razonables en el cumplimiento de estas tareas mediante la programación clásica o tradicional es algo realmente difícil y a menudo se plantearían como problemas inabarcables. Estas técnicas son efectivas cuando el problema está bien definido, como por ejemplo encontrar el camino más corto entre A y B u ordenar una lista. El *Machine Learning* permite a un ordenador aprender a realizar estas tareas por sí mismo convirtiéndose entonces en una plausible solución a

aquellos problemas clasificados como *ill-posed problems*, es decir, que no cumplen con las principales propiedades de un problema bien definido: no es seguro que una solución exista, la solución no suele ser única y los resultados obtenidos varían en función de los datos de partida [Kab08]. El Aprendizaje Automático es un método de análisis de datos capaz de automatizar la construcción de modelos. Utilizando algoritmos que iterativamente aprenden de los datos, el Aprendizaje Automático permite a los ordenadores encontrar patrones ocultos sin haber sido explícitamente programados para buscar en sitios concretos.

Arthur Samuel construyó el primer programa de Aprendizaje Automático en 1952, se trataba de un algoritmo que a medida que jugaba más a las damas se hacía más fuerte [MF90]. Sucesivamente en la historia se han ido alcanzando remarcables hitos, pero ha sido en las últimas décadas cuando se ha superado el invierno de la Inteligencia Artificial [Hen12] y realmente ha empezado una nueva revolución digital. Los motivos que justifican que en los últimos años sea posible el análisis rápido y automático de los datos con los que producir modelos capaces de analizar más cantidad de datos y de mayor complejidad, ofreciendo unos resultados más que razonables minimizando el tiempo de entrega son:

- El crecimiento en variedad y volumen de datos ha aumentado considerablemente.
- La alta capacidad de procesamiento de las máquinas se ha abaratado de manera notoria.
- El coste de almacenamiento de los datos ha disminuído cuantiosamente.

Aparte de las tareas listadas inicialmente, el Aprendizaje Automático encuentra su aplicación en muchas otras áreas:

- Coches de conducción autónoma.
- Sistemas recomendadores.
- Análisis de sentimientos en redes sociales.
- Detección de fraude.

3.1.2. Definiciones básicas

En un problema de *Machine Learning*, lo que se intenta es es estimar (aprender) los parámetros del modelo a partir de un conjunto de datos de entrenamiento, por lo que se dice que se trata de un problema inverso:

$$\text{Datos} \rightarrow \text{Parámetros del modelo}$$

Dichos parámetros dependen entonces del modelo, en algunos casos se tratará de un conjunto de pesos para cada atributo de los datos, en otros un conjunto de pares $< atributo, umbral >$ y en otros simplemente un vector de millones de números difícilmente interpretables por un humano. Por otro lado, los modelos suelen tener una serie de configuraciones que influirán drásticamente en el valor de convergencia de sus parámetros tras el entrenamiento. Estas configuraciones se conocen como hiperparámetros del modelo y permiten personalizar ciertos aspectos de la estrategia de aprendizaje.

Son dos los grupos principales de técnicas de Aprendizaje Automático: supervisado y no supervisado. En ambos, el programa intenta aprender de los datos, la diferencia es si dicho aprendizaje está supervisado por parte de un humano o no. Que el aprendizaje sea supervisado significa que

los datos utilizados en el proceso están etiquetados. Por ejemplo, si se requiere que el programa aprenda a discernir si un correo electrónico es spam o no, una opción es proporcionar un conjunto de mails etiquetados (manualmente) como spam o no spam para que él infiera las relaciones. Por otro lado, el aprendizaje puede ser no supervisado. En este caso los datos son entregados al algoritmo sin estar etiquetados con la esperanza de que por sí mismo haga visibles los patrones o agrupaciones subyacentes. En este caso el programa tiene más libertad y si se aplicase al problema de los correos electrónicos es posible que el algoritmo encontrase otras agrupaciones más allá de 'spam o no spam', como por ejemplo emails formales, familiares, de humor etc. (pero no les pondría nombre a los grupos). En este proyecto, como se verá, los algoritmos que mejor se adaptarán a las necesidades de los problemas que se intentarán resolver son del tipo supervisado.

Dentro del aprendizaje supervisado se pueden encontrar dos amplias familias de problemas a abarcar: Clasificación y regresión. De alguna manera definen también los tipos de algoritmos que existen para resolverlos. Como en este proyecto se utilizarán ambos, en la siguientes secciones se expondrán en qué consisten este tipo de problemas y cuáles son los algoritmos de los que se hará uso.

Antes de empezar a nombrarlos, es conveniente explicar algunos conceptos presentes en muchos de ellos. Se harán referencias a contenidos académicos específicos según el caso. No obstante, cabe destacar que una parte importante parte del contenido redactado se ha extraído del libro '*The Elements of Statistical Learning*'[HTF08].

En primer lugar, como se viene diciendo, en Aprendizaje Automático se intenta encontrar una función F que mapee una matriz de datos X a una salida Y de tal manera que cuanto mejor se ajuste, más potencial predictivo tendrá la función F para asignar valores a datos nuevos. Es habitual en muchos modelos (no en todos) que durante su entrenamiento se vayan haciendo cambios iterativamente en los parámetros de F mientras se va comprobando qué efecto tiene dicho cambio sobre la calidad del ajuste, para decidir cuál será el siguiente cambio. Para medir la calidad del ajuste en cualquier momento del entrenamiento se suele utilizar una función $L(X, Y)$ (*Loss function*) que puede ser entendida como el error del modelo en un instante del entrenamiento concreto [Nik11]. Si la función de pérdida tiene un valor bajo, el ajuste es bueno y al contrario. Por tanto, el problema se transforma en iteración tras iteración ir buscando la manera de minimizar L. Ahora bien, de esta manera ocurre que el algoritmo se ajustará todo lo que pueda a los datos de entrenamiento y a cualquier ruido existente en los mismos, corriendo el riesgo de producirse sobreaprendizaje. Este es un problema muy común en Aprendizaje Automático y significa que el modelo se ha sobreajustado a los datos, perdiendo la capacidad de predecir adecuadamente cualquier dato nuevo **mínimamente distinto** a los que se utilizaron en el entrenamiento. El prestigioso profesor *Andrew Ng* ilustra perfectamente este fenómeno en el capítulo 7.1 de su famoso curso abierto al público de *Machine Learning* (ver figura 3.1). El sobreaprendizaje se ve acentuado cuando se tienen demasiados atributos para un reducido número de ejemplos de entrenamiento (lo cuál intuitivamente tiene sentido). Para solucionar este problema se debe añadir un componente en adición a L que penalice configuraciones de F sobreajustadas como ocurre en los modelos de la derecha en la figura 3.1. Dicho elemento se conoce como regularización y su aceptada notación es $\lambda N(w)$, donde λ es el término de regularización, N es la norma (L_1,L_2 etc.) y w el vector de pesos correspondiente a la configuración de los parámetros de F. Con esto en cuenta, el objetivo se transforma en intentar minimizar $L(X, Y) + \lambda N(w)$.

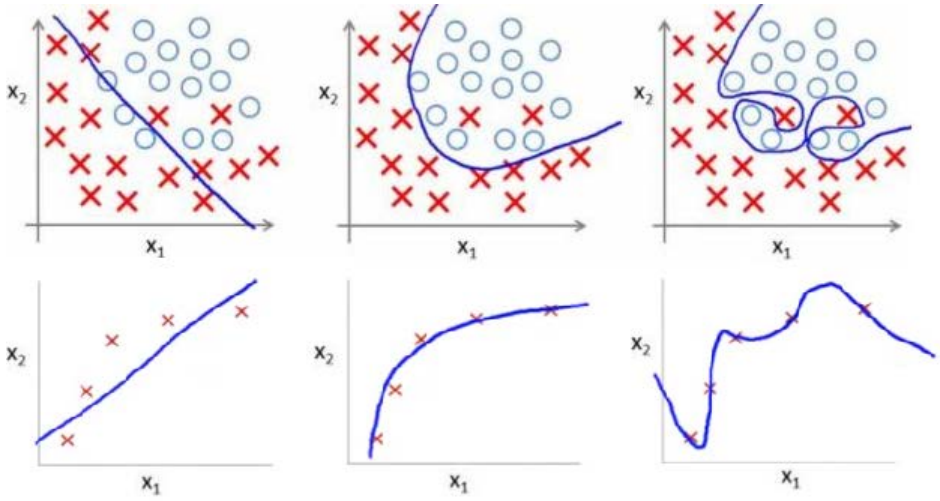


Figura 3.1: Subaprendeizaje (izquierda), aprendizaje (centro) y sobreaprendizaje (derecha) en problemas de regresión (abajo) y clasificación (arriba) [Ng08].

3.2. Clasificación

En los problemas de clasificación, se intenta predecir categorías de entre un conjunto de posibles valores. Puede tratarse de una clasificación binaria como en el caso del spam pero también pueden existir múltiples categorías, como por ejemplo países. Además, también se contempla el caso de que un mismo dato pueda pertenecer a más de una categoría. En este proyecto se trabajará con la clasificación te textos en distintas categorías, por lo que es ahí donde se centrará la explicación teórica.

El problema de la clasificación de textos

La clasificación de textos está relacionada con otra rama de la Inteligencia Artificial denominada Procesamiento del Lenguaje Natural (NLP). Ha sido un área bastante explorada ya que su dominio tiene un inmenso potencial por la cantidad de aplicaciones prácticas que tendría en el día a día de las personas pero todavía dista de ser perfecta. Uno de los problemas que se intentarán resolver en el presente proyecto es precisamente éste. En concreto, se intentará predecir a qué industria o industrias pertenece una empresa a partir de la descripción de la actividad de la misma.

Representación del texto

Para poder utilizar las descripciones es necesario transformar los textos a vectores numéricos sobre los que un algoritmo pueda realizar cálculos. La mayoría de las técnicas existentes tienen un primer paso en común: generar un diccionario o vocabulario. Este diccionario será inicialmente una lista con todas las palabras que aparecen al menos una vez en la colección de textos. Éste modelo es ampliamente usado en Búsqueda y recuperación de la información y es conocido como Modelo bolsa de palabras o *Bag of words* [Tsa12]. Cada descripción vendrá representada por un vector,

donde cada posición se corresponderá con una palabra (o *n-grama*, como se verá posteriormente) del diccionario. Ahora bien, su valor numérico puede venir determinado por distintas funciones, como se explica a continuación.

- **Conteo de palabras**: Hay varias maneras de efectuar esta representación. El conteo de palabras es una de las más sencillas y en ella, el valor que tomará cada palabra del diccionario para cada descripción vendrá determinado por la cantidad de veces que aparece en dicha descripción. Ésta manera de modelar, al ser en términos absolutos, tiene un problema, y es que dos startups con longitudes de descripción dispares pero pertenecientes a las mismas industrias tendrán representaciones desiguales. Esto tiene el potencial de producir imprecisiones en el aprendizaje. También hay que tener en cuenta que hay algoritmos más robustos que otros respecto a estas deficiencias en el conjunto de entrenamiento.

- **TF-IDF**: En cualquier caso, una buena solución consiste en elaborar una representación de las frecuencias de palabras de manera relativa y no absoluta. Dicho método es posible gracias al indicador estadístico TF-IDF (*Term Frequency – Inverse Document Frequency*) [Ram99]. Al igual que en el caso del conteo de palabras, el primer paso en éste procedimiento consiste en construir un diccionario con todas las palabras que aparecen al menos una vez en cualquiera de los textos. Ahora bien, la diferencia entre ambas técnicas radica en que en éste caso, el vector representativo de un texto no contiene frecuencias absolutas, sino una medida de cuán relevante es cada posible palabra para ese texto teniendo en cuenta la entera colección. Éste valor aumentará proporcionalmente si la frecuencia de la palabra es alta en la descripción, pero se compensará según la frecuencia de la misma teniendo en cuenta todas las descripciones, de tal manera que el cómputo es sensible al hecho de que algunas palabras son más comunes que otras.

  Su cálculo es el producto de los dos indicadores siguientes:

  - **TF**: *Term Frequency*, mide la frecuencia de un término dentro de un texto, teniendo en cuenta que la longitud medida como número de palabras de los mismos puede variar (es decir, normalizando):

$$TF = \frac{Frecuencia \quad de \quad la \quad palabra \quad en \quad el \quad texto}{Longitud \quad del \quad texto} \tag{3.1}$$

  - **IDF**: *Inverse Document Frequency*, mide la importancia de un término analizando si es muy frecuente o no en toda la colección de textos. Supone que la palabra *the* (frecuente en todos los textos) debe ser menos importante que *healthcare* (menos frecuente):

$$IDF = \ln\left\{\frac{N\acute{u}mero \quad de \quad textos}{N\acute{u}mero \quad de \quad textos \quad conteniendo \quad la \quad palabra}\right\} \tag{3.2}$$

Clasificadores

El problema en cuestión consiste en llevar a cabo una clasificación multi-etiqueta, ya que habrá que etiquetar con varias industrias a cada empresa (las industrias no son mutuamente excluyentes). De los clasificadores implementados en *Scikit-learn*, que como se verá en la sección 4.1.2 será una herramienta muy utilizada a lo largo del proyecto, se probarán los siguientes estimadores:

- **RidgeClassifier**: La principal característica de este clasificador lineal es que utiliza como término de regularización el método de Tikhonov-Miller (norma L_2) con el objetivo de superar algunas de las limitaciones del método de los mínimos cuadrados ordinarios a la hora de abordar problemas inversos no bien definidos como la clasificación multi-etiqueta de textos. Concretamente, en problemas con potencialmente muchos atributos y grados de libertad como este, destaca su capacidad de hacer más que aceptables generalizaciones sin necesidad de ejecutar previamente un algoritmo de selección de atributos, evitando el riesgo de acabar con soluciones óptimas pero específicas para un conjunto de entrenamiento concreto. En conclusión, el principal punto a favor de éste algoritmo es su fortaleza frente al sobreaprendizaje [GHO99].

- **Perceptron**: Data de 1950 y fue de las primeras implementaciones basadas en modelos neuronales. De hecho, puede ser considerado como el algoritmo pionero que sentó las bases sobre las cuáles se sostienen muchos de los potentes modelos de aprendizaje automático de hoy en día. Simple y elegante, realiza un aprendizaje supervisado de una función lineal ajustando un vector de pesos. La elegancia y sutileza de este modelo matemático queda patente en su prueba de convergencia [Col12]. La implementación por defecto en *scikit-learn* no necesita de una tasa de aprendizaje para controlar la convergencia, sólo actualizada los pesos cuando ocurren errores y (a diferencia del *RidgeClassifier*) no está regularizado.

- **PassiveAgressiveClassifier**: Este modelo pertenece a una familia de algoritmos utilizados en aprendizaje de sobre datasets de gran escala. Se parece al *Perceptron* en el sentido de que a priori no requiere de una tasa de aprendizaje, pero se diferencia de él en qué sí tiene que definirse un parámetro de regularización para penalizar los errores durante la fase de aprendizaje [CDKSS06].

- **KNeighborsClassifier**: Es posiblemente el clasificador basado en instancias más popular. La principal característica de este tipo de modelos es el hecho que su fase de aprendizaje tiene un coste mínimo. De hecho, dicha fase consiste principalmente en almacenar todas las instancias disponibles. Esta aparente ventaja tiene su contra parte a la hora de elaborar predicciones, ya que el algoritmo en su implementación más básica tiene que medir similitudes (a menudo valiéndose de la distancia euclídea) entre la instancia a predecir y las instancias almacenadas. Después, se establece la clase como aquella más común de entre las K instancias más afines. Por tanto, según va creciendo el número de ejemplos en el dataset, más recursos consumirá el proceso de predicción. A pesar de su simpleza y fácil implementación, a menudo resulta un algoritmo eficaz en la práctica, por lo que es ampliamente utilizado en la industria.

- **DecissionTree**: Los árboles de decisión intentan inferir a partir de los datos conjuntos de reglas simples (del tipo *if-else*). De manera intuitiva se puede decir que buscan recursivamente cuáles son los atributos que *mejor* dividen los datos atendiendo al *output*. De tal manera que una vez que el modelo ha sido entrenado, la predicción de una instancia tiene lugar simplemente aplicándole sucesivamente las reglas generadas hasta llegar a un nodo hoja donde se le asignará un valor. Existen varias implementaciones (*ID3, C4.5, C5.0, CHAID, MARS...*) pero en este proyecto se utilizará *CART* porque es la que viene en el paquete de *scikit-learn* y además vale tanto para clasificación como para regresión. Los aspectos positivos más importantes de este algoritmo son:

  - Es capaz de aprender relaciones no lineales entre los datos.

- Detectar con eficacia cuáles son los atributos más relevantes y su rendimiento no se ve afectado por la inclusión de otros que no lo son.
- Son 'cajas blancas', es decir, son capaces de mostrar los razonamientos que ha aprendido mediante un grafo (en la figura 7.13 de la sección 7.1.5 se mostrará un ejemplo).
- Apenas requieren preparación de los datos (escalados, normalizaciones, estandarizaciones, codificaciones de atributos nominales etc.).

Por su parte, algunos inconvenientes son que a veces pueden crear modelos demasiados complejos que no generalizan bien (sobreaprendizaje). Relacionado con esto, pequeñas variaciones en los datos de entrenamiento pueden acabar en la construcción de árboles potencialmente diferentes [HTF08].

- **RandomForestClassifier**: Pertenece a la familia de los métodos ensambladores, bastante populares en la industria [CG16]. Este tipo de meta-estimadores permiten la creación de modelos precisos mediante la combinación de varios modelos débiles (normalmente árboles de decisión) entrenados con *ligeras* variaciones entre los mismos. Se usan tanto para clasificación como regresión. En la sección 3.3.2 se les dará una mayor cobertura.

- **LinearSVC**: Método de aprendizaje supervisado perteneciente a la familia de máquinas de soporte vectorial. Esta familia de estimadores está compuesta en esencia por clasificadores binarios que se pueden transformar tanto en problemas de clasificación multietiqueta como en regresión. Partiendo de un conjunto N-dimensional de datos, generan un hiperplano de $N-1$ dimensiones que intenta separar los datos en dos de la mejor manera posible. Hasta aquí no se diferencian de otros muchos modelos. Lo que les hace únicos es la utilización del 'truco del *kernel*' para conseguir dicha separación. Resultaría muy difícil explicar claramente en qué consiste dicha técnica en pocas líneas, por lo que se invita al lector interesado a consultar la bibliografía [MMRTS01]. A nivel intuitivo, dicho método está relacionado con la ejecución de funciones de distancia (*kernels*) sobre las instancias permitiendo entre otras cosas convertir un problema no lineal a uno lineal (dependiendo del tipo de *kernel* que se utilice). La clave está por tanto en cómo estas funciones calculan las similitudes. *LinearSVC* utiliza un *kernel* lineal, que es el más sencillo y el primero que se suele probar, pero también existen *kernels* gausianos/de base radial, capaces de aprender relaciones de mayor complejidad.

- **SGDClassifier**: Este clasificador aprende un modelo lineal (como *LinearSVC* o una regresión logística) pero utiliza el método del gradiente estocástico para optimizar la función de coste al entrenar, de manera similar al el algoritmo de propagación hacia atrás comúnmente utilizado en las redes de neuronas artificiales. Además, *SGDClassifier* utiliza una versión optimizada de este método (*mini-batch*) de tal manera que permite el aprendizaje de grandes datasets sin consumir tantos recursos como el método del gradiente convencional [LZCS14]. Esta característica le hace una opción ciertamente eficiente que ha ganado popularidad en la última década a pesar de llevar en la comunidad desde hace bastante tiempo debido a que en los últimos años la cantidad de datos a crecido más rápido que la velocidad de los procesadores [Bot10]. El método del gradiente estocástico ha sido satisfactoriamente probado con datos de naturaleza dispersa típicamente encontrados en clasificación de textos y procesamiento del lenguaje natural, por lo que parece un buen candidato para el problema.

- **MultinomialNB**: Éste modelo también se ha mostrado eficaz varias veces en dominios de clasificación de textos [MN98] pero el método en sí pertenece a una familia de técnicas to-

talmente distintas. Consiste en una implementación del clásico *Naïve Bayes* adaptada para datos que siguen una distribución multinomial. Según el manual de *Scikit-learn*, aunque este tipo de distribución normalmente requeriría una representación del texto como un vector de números enteros (conteo de frecuencias de palabras), es perfectamente posible que en la práctica conteos fraccionales como el *tf-idf* puedan dar buenos resultados [Sci14c].

- **BernoulliNB**: De la misma familia que el anterior pero en este caso se asume que los datos siguen una distribución multivariada de *Bernoulli*. En la práctica, la principal diferencia es que mientras que en esta versión se penaliza la no ocurrencia de un término, en el anterior modelo se ignoraría, por lo que los resultados pueden variar sustancialmente. Además, que uno acabe funcionando mejor que el otro dependerá en última instancia del tipo de dataset, siendo *BernoulliNB* el recomendado cuando los documentos a clasificar son cortos [MN98].

La mayoría de los clasificadores listados están diseñados inicialmente para problemas en los que la clase a predecir puede tomar distintos valores pero cada instancia sólo puede estar asociado a uno de ellos. El problema que se abordará requerirá que un mismo dato pueda pertenecer a varias clases a la vez, por lo que muchos de los estimadores listados no se podrán utilizar directamente. La manera en la que comúnmente se soluciona este problema es, en vez de entrenar un sólo estimador, entrenar tantos como clases existan. De esta manera, si se tuviesen 10 clases posibles, y cada dato pudiese pertenecer a más de una, habría que entrenar un estimador binario por cada clase que se encargase exclusivamente de predecir si el dato pertenece o no a esa clase. De esta manera resulta sencillo sumar a posteriori las salidas de cada uno de los estimadores. Esta técnica es conocida como *One versus All* ó *One versus The Rest* [Sci14d].

3.3. Regresión

A continuación se explicarán los algoritmos que se probarán para afrontar los problemas de regresión que surjan en el proyecto y por qué. Se les ha clasificado en dos: los estimadores (sección 3.3.1) y los meta-estimadores (sección 3.3.2).

3.3.1. Estimadores

- **SVR**: Será interesante probar las máquinas vectoriales con distintos *kernels* también en este tipo de problemas.
- **Lasso**: Este estimador lineal se caracteriza principalmente por utilizar la norma *L1* como regularización para evitar el sobreaprendizaje. Suele ser útil para hacer cribas de atributos ya que cuando λ es suficientemente grande en $\lambda L_1(w)$, muchos de los pesos w_i tienden a 0 (resultando en vectores más dispersos e interpretables que si se usase la norma L_2). Se utilizará porque se entrenarán modelos con un potencialmente elevado número de atributos que posiblemente estén introduciendo ruido en los datos.
- **RidgeRegression**: Es exactamente la misma idea que se explicó en el *RidgeClassifier* de la sección 3.2 (lineal y hace uso de la norma L_1) pero utilizado en problemas de regresión.
- **ElasticNet**: También aplica para problemas lineales como *Lasso* y *Ridge* pero utiliza como regularizador combinaciones de las normas L_1 y L_2, respectivamente. Por su parte, *ElasticNet* es más estable que *Lasso* cuando se trata con un número considerable de atributos de

importante correlación. Como es de esperar, la minimización de las normas L_1 y L_2 a la vez conlleva una complejidad extra importante pero que se resuelve de manera eficiente mediante un descenso coordinado o de gradiente [ZH03].

3.3.2. Meta-estimadores

Dentro del Aprendizaje Automático, estos métodos se caracterizan por hacer uso de otros algoritmos de Aprendizaje Automático como análisis de principales componentes para aprender colecciones de predictores, ya sean estos clasificadores o regresores. Todos los integrantes de esta familia de metaalgoritmos hacen una apuesta en común:

> 'Un conjunto de predictores dará mejor resultado que uno sólo.'

Existe una famosa analogía que ejemplifica la motivación detrás de los modelos ensambladores: cualquier fiel espectador del programa *¿Quién quiere ser millonario?* se habrá dado cuenta de un curioso fenómeno que tiene lugar cuando el concursante que no tiene idea de qué responder decide utilizar uno de sus dos comodines (llamada o comodín del público). A menudo, el amigo 'experto' al que el concursante llama se equivoca, mientras que si toma el voto mayoritario de la audiencia tras utilizar el comodín del público acertará prácticamente en todos los casos. De alguna manera, una agrupación de cientos de predictores 'no expertos' resulta más efectiva que uno que sí lo es actuando por sí mismo. Este fenómeno ha sido profundamente estudiado y una de las explicaciones más aceptadas es que existe un ruido idiosincrático asociado a cada juicio individual, y el hecho de tomar la media de todos ellos hace que se cancele dicho ruido [YSLD12]. La idea subyacente de los métodos ensambladores es por tanto formar un robusto predictor a base de juntar varios débiles.

A menudo, un ensamblador realiza una predicción combinando cada una de las predicciones de los estimadores que lo integran, a los cuáles va asignando pesos. Estos estimadores, a su vez, pueden ser del mismo o distinto tipo. Esta característica resulta especialmente útil cuando se afronta un problema y no se tiene idea de qué modelo es el adecuado, construyendo un ensamblador combinando redes de neuronas con árboles de decisión y máquinas de soporte vectorial ya que durante la fase de entrenamiento se ponderarán adecuadamente los pesos de los estimadores según la cantidad de errores que cometan. Por otro lado, también se podrán utilizar varias instancias del mismo predictor pero haciendo algunas variaciones de el entrenamiento de cada una de ellas, de tal manera que se facilite la adaptación global del modelo al dataset.

Los meta-estimadores son algoritmos con cierta complejidad computacional y hacen un uso intensivo de CPU. No obstante, una ventaja práctica de esta clase de predictores es que su naturaleza algorítmica les hace ser procesos computacionalmente paralelizables en entrenamiento y evaluación. Afortunadamente, las excelente implementación de *Scikit-learn* aprovecha esta característica y permite hacer uso de todos los procesadores que tenga la máquina sobre la que se ejecute.

Se pueden diferenciar dos familias dentro de los métodos de ensambladores atendiendo a la manera de estructurar las colecciones de predictores:

- **Métodos de comité**: Se construyen varios predictores de manera independiente y se toma como salida una media de las predicciones de cada uno de ellos. A menudo, el estimador base que se utiliza es un árbol de regresión o clasificación. Aplicados individualmente, los árboles se muestran como algoritmos muy flexibles capaces de aprender relaciones no lineales entre los datos de entrada. No obstante, a menudo padecen de errores grandes en sus predicciones y son muy inestables, que en términos de Aprendizaje Automático se traduce en alta varianza

y sobreaprendizaje. Esto último significa que pequeños cambios en las instancias de entrenamiento pueden desembocar en resultados altamente diversos. Los métodos ensambladores aprovechan esta característica aleatorizando las distintas partes del proceso de construcción del árbol para cada uno de sus árboles, desde el dataset hasta los atributos, de tal manera que los errores graves específicos de cada predictor serán ignorados por el comité (reducción de la varianza) mientras que el modelo general se ajustará con gran precisión. Aunque estos métodos suelen superar siempre los resultados de los árboles individualmente, es conveniente recalcar que se pierde uno de los atractivos de los árboles: que son fácilmente interpretables. Concretamente, dentro de este grupo podemos diferenciar los siguientes tipos de algoritmos según la manera en la que se crean los árboles:

- **Bagging** o ***B**ootstrap **Agg**regat**ing***, cuya idea principal es generar los distintos subsets para cada estimador mediante la técnica *bootstrapping*, consistente en la realización de muestreos aleatoriamente uniformes y con reemplazo (los datos pueden repetirse en cada muestra) sobre el dataset original [Efr79]. No obstante, en ocasiones el uso de esta técnica no es suficiente para garantizar la diversidad del modelo. En ocasiones ésta no se ve limitada simplemente por los muestreos de los datos de entrenamiento sino por el proceso de construcción de los árboles en sí. Lo que implica que en *Bagging* a veces los árboles generados sean similares (correlación entre estimadores), limitando la diversidad del modelo global. Por ello, este tipo de meta-algoritmo tiende a funcionar mejor con predictores complejos como por ejemplo árboles con mucha profundidad. En definitiva, es buena idea usarlo cuando el estimador base tiene alta varianza (ya que la disminuye) y poco sesgo (ya que no lo disminuye tanto) [Bre94].
- **Selvas Aleatorias**, más conocido como *Random Forests* intenta solventar el problema de la diversidad limitada del *Bagging* introduciendo aleatoriedad en el proceso de construcción del árbol. Es decir, además de utilizar *bootstrap*, se forzará a que cada árbol sólo se pueda ramificar en un subset determinado aleatoriamente a partir del conjunto de atributos original. Dado que en este caso la aleatorización es todavía mayor, es probable que se produzcan ligeros aumentos en el sesgo del bosque (respecto a abordar el problema con un sólo árbol) pero este contratiempo suele contrarrestarse satisfactoriamente con la considerable disminución de la varianza.
- **Árboles Extra-Aleatorios**, o *Extremely randomized trees* [GEW05]. En este tipo de meta-estimador la aleatoriedad se lleva al siguiente nivel en lo que respecta a las ramificaciones que se producen. De la misma manera que en las *Selvas Aleatorias*, un subconjunto de los atributos candidatos es utilizado, pero a continuación, en lugar de establecer el umbral que mejor discrimine los datos directamente, se probarán distintos umbrales **de manera aleatoria** para cada atributo seleccionando definitivamente el que mejor resultados haya dado. Este extra de aleatoriedad contribuye a reducir todavía más la varianza, pero aumenta el sesgo del estimador.

- **Métodos de *Boosting***: Estos algoritmos también tratan de construir un robusto predictor a partir de un conjunto de estimadores más débiles. No obstante aquí el enfoque es distinto, ya que los distintos regresores o clasificadores no se construyen en paralelo, independiente del resto. Se combinan en serie de tal manera que el segundo se creará a partir de las estimaciones del primero y así sucesivamente. La motivación que existe detrás de este diseño es simple: corregir los errores del modelo anterior. Así, se añadirán estimadores uno tras otro hasta que

se alcance un límite predefinido o el error de entrenamiento sea nulo. Esta categoría suele ser especialmente efectiva cuando se combinan varios predictores simples. Esto son, estimadores algo mejores que un predictor aleatorio, como por ejemplo árboles de decisión de profundidad reducida. Si se utilizasen modelos más complejos (como es recomendable hacer en *Bagging*) se dificultaría el aprendizaje de la salida de unos estimadores por parte de los que vienen a continuación el cadena.

- **AdaBoost** Fue la primera implementación de esta familia de meta-estimadores que encontró una estrategia especialmente efectiva para ir asignando los pesos a cada instancia de entrenamiento de tal manera que los datos que se estimasen peor (predicciones difíciles) tuviesen mayor prioridad que los que se estimasen mejor (predicciones fáciles). Una vez que el entrenamiento y ensamblaje secuencial de los estimadores han concluido se podrá proceder con las predicciones de datos reales de tal manera que todos los componentes del ensamblador votarán siendo asignados con un peso que variará en función de lo bueno o malo que fue su rendimiento durante el aprendizaje [Dru97].

- **Gradient Boosting** es una generalización muy potente de los métodos de *Boosting* que permite optimizar el modelo ofreciendo la posibilidad de minimizar varias funciones de pérdida. Estas funciones tienen que ser derivables ya que este algoritmo se diferencia del *Adaboost* en que mientras en éste se identifican las predicciones difíciles mediante pesos altos, aquel utiliza la técnica del descenso del gradiente. En ésta técnica se mide la dirección de aumento del error para cada dato mal estimado y se modifican los parámetros de estimación global para moverse en el sentido contrario en la búsqueda de un mínimo local de una función de pérdida [Fri01].

3.4. Técnicas de evaluación de modelos

3.4.1. Para clasificación

Como se ha podido observar en la sección 3.2, existe una gran variedad de algoritmos, y además cada uno de ellos cuenta con su propio conjunto de parámetros que afectarán a su rendimiento y capacidad generalizadora. Por lo tanto resulta crucial elegir un conjunto de métricas en las que fijarse para evaluar todas las configuraciones de modelos que se deseen incluir en los experimentos. Aunque existen otras muchas métricas perfectamente válidas para evaluar clasificadores [14c], estas son las principales métricas para evaluar los estimadores encargados de resolver problemas de clasificación multi-etiqueta.

- **Precision**: También conocido como exactitud, mide la habilidad de un clasificador de no etiquetar como positiva una muestra que es negativa (ver figura 3.2).

- **Exhaustividad**: También llamado sensibilidad, mide la completitud de la clasificación. Una baja exhaustividad indica muchos falsos negativos (ver figura 3.2).

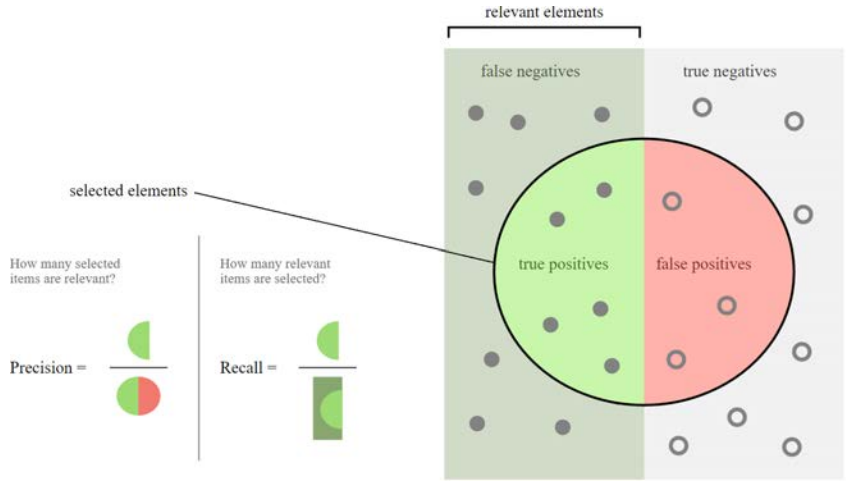


Figura 3.2: Explicación gráfica de precisión y exhaustividad. Fuente: *Wikipedia*[14f]

- **F1-score**: También conocido como F-score a secas, hace un balance de las dos métricas anteriores:

$$F1 - score = 2 \times \frac{Precisión \times Exhaustividad}{Precisión + Exhaustividad}$$

3.4.2. Para regresión

Las métricas a evaluar serán las siguientes. En todas ellas, $n_{instancias}$ indica el número de ejemplos que tiene la muestra sobre la que se hace la evaluación, y es el valor real de la salida e $\hat{y}$ se corresponde con el valor estimado por el modelo :

- **Error absoluto medio**, mide la diferencia media que existe entre los valores predichos y los reales.

$$EAM\left(y, \hat{y}\right) = \frac{1}{n_{instancias}} \sum_{i=0}^{n_{instancias}-1} |y_i - \hat{y}_i|$$

- $\boldsymbol{R^2}$ [01], es una de las claves a la hora de evaluar la salida de cualquier modelo de regresión. Se interpreta como la proporción de la varianza de la variable dependiente que es predecible a partir de la variable independiente. Se conoce como el coeficiente de determinación y en términos generales proporciona una medida que indica como de buenas serán las predicciones del modelo. Su valor máximo es 1 (lo mejor) y no está acotado inferiormente. Es relevante mencionar que un modelo que siempre predice el valor esperado de y independientemente de los atributos, tendrá un $R^2 = 0$.

$$R^2\left(y, \hat{y}\right) = 1 - \frac{\sum_{i=0}^{n_{instancias}-1} \left(y - \hat{y}\right)^2}{\sum_{i=0}^{n_{instancias}-1} \left(y - \bar{y}\right)^2}$$

donde:

$$\bar{y} = \frac{1}{n_{instancias}} \sum_{i=0}^{n_{instancias}-1} y_i$$

- **Varianza explicada** [Ken83], mide la proporción de la varianza total existente en los datos que el modelo matemático es capaz de explicar y viene dada por la siguiente fórmula, siendo y el valor del *output* real e $\hat{y}$ el valor estimado:

$$varianza\_explicada\,\{y, \hat{y}\} = 1 - \frac{Var\,\{y, \hat{y}\}}{Var\,\{y\}}$$

  Cuanto más alto sea el valor de la métrica, mejor. Como máximo puede valer 1 y no está acotada inferiormente. En su cómputo se utilizada la varianza sesgada, mientras que en el de R^2 se usa la no sesgada.

3.5. Combinación de distintas evaluaciones

Si se entrena cualquier modelo con la totalidad del dataset y a continuación se computa cualquiera de estas métricas sobre las salidas del estimador y las salidas reales en el dataset, no se estará midiendo la capacidad real de generalizar/predecir del modelo, ya que se le estará examinando haciéndole preguntas cuya respuesta ha visto durante el entrenamiento. La manera correcta de evaluar consiste en hacerle preguntas totalmente nuevas. Si se computan las métricas exclusivamente para estas preguntas, entonces sí que su puntuación será un buen indicador de la capacidad generalizadora del estimador.

Es habitual utilizar aproximadamente un 75 % de las instancias disponibles para entrenar el modelo (*train set*) y el 25 % restante para evaluarlo (*test set*). Será con este 25 % último con el que se computarán todas las métricas deseadas que servirán para hacerse una idea de la capacidad generalizadora del modelo, tanto en los problemas de clasificación como en los de regresión.

Por otro lado, la evaluación conseguida de esta manera todavía dista de ser precisa/óptima porque es muy posible que los resultados varíen en función de con qué tipo de instancias se entrene y con cuáles se evalúe. La solución habitual para este problema consiste en utilizar validación cruzada [Koh95]. Esta técnica consiste en primero fijar un número k de iteraciones o *folds*. Después se generan y evalúan los k modelos, cada uno con un conjunto distinto de *train* y *test sets* para finalmente hacer la media de las métrica deseadas calculadas individualmente para los modelos generados. En el caso de este proyecto, se aprovechará un tipo de evaluación implementada en *scikit-learn* llamada *Cross_val_predict*. Esta función conlleva la validación cruzada pero además de computar las medias de las métricas de los k modelos, hace también una 'media' de las predicciones de cada una. Esta manera de generar estimaciones cruzadas permite poder observar las predicciones del modelo evaluado sin tener que entrenar más de una vez, fusionando varias líneas de código en una sola. Será especialmente útil porque es interesante poder observar predicciones en las salidas de consola personalizadas durante el proceso de evaluación.

3.6. Metodologías

Está más que comprobado que una de las claves del éxito del desarrollo de software es el adecuado uso de metodologías, especialmente en la industria. Debido a la propia naturaleza del Aprendiza-

je Automático, el desarrollo de sistemas que hacen uso de él está diferenciado del desarrollo de aplicaciones más 'tradicionales'. Por ello, existen metodologías específicamente diseñadas para este tipo de proyectos. Dado que este trabajo está muy relacionado con *Data Science*, se ha tomado la decisión de seguir una de estas metodologías.

El prestigioso blog *KDnuggets: Data Mining, Analytics, Big Data, and Data Science*[1] dirigido por *Gregory Piatetsky-Shapiro* ha realizado sucesivas encuestras en los últimos años acerca de qué metodologías son las más usadas en proyectos de estas características (ver figura 3.3). Llama la atención que los resultados de la encuesta en 2014 apenas han variado respecto a los de 2007.

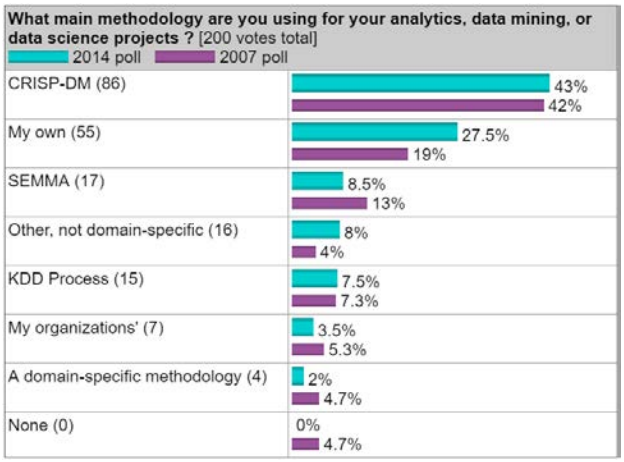


Figura 3.3: Resultados de una encuesta de *KDnuggets* acerca de metodologías utilizadas en proyectos *Data Science* en 2014 [14e]

Tras consultar el pequeño estudio se decide que en este proyecto se utilizará la metodología *Cross Industry Standard Process for Data Mining*, comúnmente conocida como *CRISP-DM* [C00]. Los motivos son:

- Es la más utilizada con diferencia en la industria. Esto es, especialmente al hablar de metodologías, una señal a tener muy en cuenta.
- Ha sido numerosas veces aclamada y reconocida como un estándar para el desarrollo de proyectos intensivos en *Data Mining* y descubrimiento de conocimiento [MMS09].
- Pone el entendimiento de negocio (como se verá en el siguiente apartado) en el centro y al frente desde el inicio del proyecto.
- El modelo de decisión central planteado se sustenta sobre algo más que mejorar ciertas métricas. El modelado de decisiones ayuda a que los proyectos de *Data Analysis* se centren en mejorar la manera en la que los negocios actúan hoy en día a la vez que suponen un fantástico activo para la planificación del despliegue.

[1] `www.kdnuggets.com`

CRISP-DM

En la figura 3.4 se muestra un grafo con las fases que componen dicha metodología. Como se puede apreciar, no se trata de un modelo lineal sino que las fases forman un continuo ciclo de actividad analítica en el que está permitido y de hecho es muy común iterar repitiendo unas con los resultados de otras:

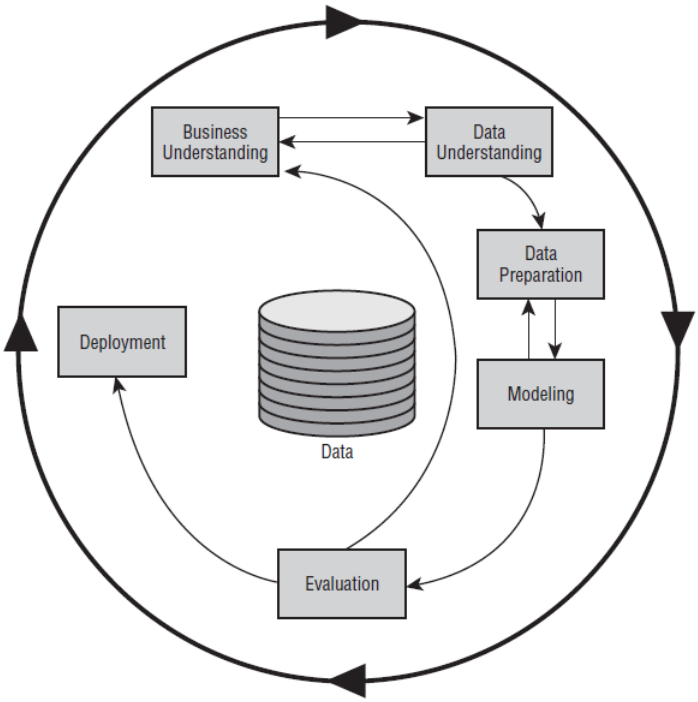


Figura 3.4: Fases de la metodología *CRISP-DM*

A continuación se describe brevemente en qué consiste cada una de las fases.

1. **Entendimiento del negocio** Obtener un entendimiento claro de cuál es el problema que se va a resolver, cómo impactará en la organización y los objetivos necesarios para resolverlo. Esto implica analizar cuáles son los objetivos y requisitos del problema desde una perspectiva de negocio a partir de los cuáles se define el proyecto.

2. **Entendimiento de los datos** Obtener, inspeccionar, describir y evaluar los datos. Además, también se pueden ir formando hipótesis acerca de la información todavía oculta.

3. **Preparación de los datos** Convertir los datos de su fuente original a el formato deseado. No sólo se trata de hacer una limpieza (siempre obligatoria) sino que en esta fase también se deben tomar decisiones acerca de cuáles serán los atributos o campos que se incluirán en el dataset final.

4. **Modelado** Seleccionar y utilizar técnicas matemáticas y algoritmos que capaces de extraer el conocimiento deseado de los datos. Es común en esta fase el uso te técnicas de búsqueda de la configuración óptima de estos modelos.

5. **Evaluación** Comprobar cómo de bueno o malo es el modelo. También se debe conectar con la primera fase en el sentido de asegurarse de que el modelo construido cumple con los objetivos definidos inicialmente.

6. **Despliegue** Si el objetivo era extraer conocimiento de los datos, generar un informe puede ser suficiente, pero si el objetivo era integrar el modelo con un sistema de producción requerirá más trabajo.

En el presente proyecto se utilizarán técnicas de Aprendizaje Automático/*Data Mining* en distintas fases, correspondiéndose cada una de ellas con problemas que se deben tratar obligatoriamente por separado. Por tanto, se aplicará esta metodología independientemente en cada parte y existirá una última fase donde habrá una convergencia, a modo de despliegue global. Para poder operar de esta manera se prestará especial atención a la fase de 'Entendimiento del negocio' de cada subproblema, donde se establecerán algunos de los objetivos del proyecto (sección 1.3) como objetivos específicos del dicho subproblema.

Por otro lado, para abordar algunos de los subproblemas del proyecto se utilizarán técnicas que aplicarán conocimiento experto y harán uso de sistemas de reglas. Para estos casos, donde el Aprendizaje Automático no estará presente, no se utilizará *CRISP-DM*. De hecho no se aplicará formalmente ninguna metodología concreta ya que dichos subproblemas no tendrán demasiada carga y serán resueltos exclusivamente por el autor (no será necesario coordinar un equipo de desarrolladores y analistas). No obstante, la manera de abordarlos será mediante las clásicas fases de definición del problema, diseño de la solución, evaluación y despliegue.

Capítulo 4

Herramientas

4.1. Lenguaje y librerías

Hay que elegir un lenguaje de programación en el que escribir las funcionalidades del sistema. Dicho lenguaje debe contar con librerías de manejo de grandes cantidades de datos, análisis estadísticos y aprendizaje automático. Además, aparte de poder realizar experimentos puntuales, el sistema debería ser escalable y poder ser implementado como para ser usado en producción. Primero se comentará brevemente cuáles son los 4 lenguajes que más se utilizan actualmente en el mundo del *analytics*, *data mining* y *data science* y finalmente se justificará la elección de uno de ellos para el presente proyecto.

4.1.1. Comparativa

En 2013 *KDNuggets* lanzó una encuesta para averiguar cuáles era los lenguajes más utilizados en este dominio [13a]. La figura 4.1 muestra cuáles fueron los más populares.

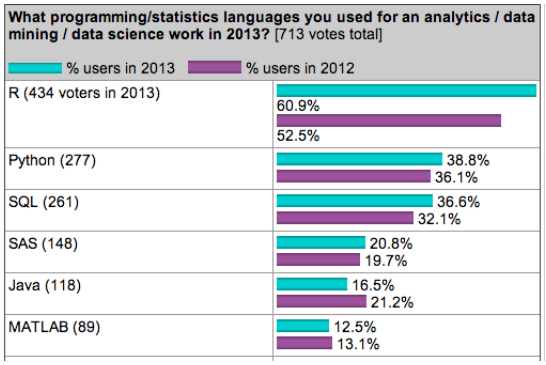


Figura 4.1: Resultados de una encuesta de *KDnuggets* acerca de lenguajes de programación populares en *Data Science* en 2013 [13a]

Los resultados que se observan en la figura 4.1 son muy similares a los de 2012, con un uso intensivo de *R* y *Python* para análisis de los datos y *SQL* para la gestión de los mismos. Este último no tiene cabida en la comparativa ya que su uso complementa al resto de las opciones y no las sustituye. A continuación se justifica se efectuará una comparativa de estas herramientas y se discutirá cuál es la mejor alternativa:

- **MATLAB/Octave** (licencia comercial/libre): Son excelentes representando y trabajando con matrices, por lo que son muy apropiados para optimizar procesos intensivos en álgebra lineal. Muy populares en el mundo académico para explorar y experimentar dado que además cuentan con muchas funcionalidades de análisis de datos y algoritmos. En su contra, su sintaxis es algo tediosa, tiene tiempos de ejecución lentos y no cuenta con entornos de desarrollo muy amigables. Para finalizar, no es fácil integrarlo en un sistema de producción.

- **Java**: Lenguaje de programación de propósito general y orientado a objetos cuya característica principal es que está diseñado para tener las mínimas dependencias de implementación posibles (*write once, run anywhere* [98]). Es un lenguaje con unas características que se adaptan perfectamente a los requisitos de un sistema de producción, y también hay que tener en cuenta que el autor de este proyecto tiene mucha experiencia con él. Contextualizando el lenguaje en este proyecto, la principal librería de Aprendizaje Automático de Java es *Weka*, que es una fantástica herramienta para aprender y testear algoritmos clásicos y que además cuenta con una interfaz gráfica de usuario bastante amigable. No obstante, *Weka* no es fácilmente integrable, su comunidad es pequeña respecto al resto de opciones, sus algoritmos no representan el estado del arte del Aprendizaje Automático y su gestión de memoria deja que desear (en ocasiones teniendo el usuario que aumentar el espacio del montículo[1] manualmente), por lo que a pesar de ser estupendo para un uso académico, no es apropiado para un sistema de producción que trabaje dinámicamente con datasets de gran tamaño.

- **SAS**: Ha liderado el mercado de *Analytics* durante mucho tiempo. Es un *software* comercial con multitud de funciones estadísticas y tiene una interfaz gráfica de usuario amigable para garantizar una curva de aprendizaje mínima. Aparte de que es una opción bastante cara, no refleja el estado del arte del Aprendizaje Automático al no tener una comunidad de desarrolladores abierta y las funcionalidades que incluye no son personalizables. Su utilización está en decadencia. No se utilizará en el proyecto.

- **R**: Se puede considerar como la contrapartida de *SAS* en código abierto. Es especialmente utilizado en el mundo académico para la investigación. Tiene una comunidad bastante grande y activa en *Stack Overflow*. *CRAN* es un gigante repositorio de paquetes a los que se puede acceder y contribuir de manera directa. Cuenta con completas herramientas para la visualización de gráficos. No obstante, hay que mencionar que se trata de un lenguaje desarrollado por y para estadísticos. Esto tiene la desventaja de que al centrarse tanto en ofrecer una inmensa cantidad de funcionalidades para correr experimentos de diversa índole, se olvida en cierta manera del rendimiento de la máquina, haciendo de él un lenguaje con finalidades más exploratorias que productivas. Además, tiene una curva de aprendizaje importante si se compara con otros lenguajes.

[1] Comúnmente conocido como *heap*, es una región de la memoria de un ordenador donde se almacenan variables dinámicas. Crece y disminuye según se crean o liberan estas variables mientras que el programa se está ejecutando.

- **Python**: Lenguaje de código abierto basado en *scripts* cuya popularidad se ha incrementado muchísimo en los últimos años, llegando a tener un activo ecosistema de usuarios y desarrolladores. Hoy en día cuenta con librerías con funciones muy potentes para realizar prácticamente cualquier operación estadística, construcción de modelos y gestión de datos. Sus funcionalidades cubren el espectro del estado del arte del Aprendizaje Automático y análisis de datos, pero son inferiores en número respecto a *R* (la distancia se estrecha cada vez más). Lo más ventajoso de este lenguaje respecto a su principal competidor, *R*, es que al ser un lenguaje de programación de propósito general permite poder explorar/experimentar, testear y finalmente integrar en un sistema de producción todo con la misma herramienta. Por todo ello será el lenguaje elegido.

4.1.2. Python

Python es un lenguaje sencillo y fácil de aprender (lo que puede ser más difícil es aprender a utilizar determinadas librerías). Es posible hacerse una idea de las buenas prácticas de estilo de este lenguage consultando lo que se conoce como *Zen of Python* ejecutando desde cualquier *script* la línea `importthis` [91], mostrándose por pantalla los 19 principios del lenguaje como se muestra en la figura 4.2.

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Figura 4.2: Principios de programación del lenguaje *Python*

Es además muy dinámico, lo que lo convierte en un lenguaje apropiado para un desarrollo interactivo y prototipado rápido. Por otro lado, es ampliamente usado en el renovado mundo del Aprendizaje Automático y estadística por las excelentes librerías desarrolladas por terceros con las que cuenta; sin olvidarse de que el hecho de ser de propósito general (no es el caso de *R* ni de *Matlab*/*Octave*) también ayuda. *Python* cuenta principalmente dos versiones (2.x y 3.x). Ambas versiones difieren ligeramente en su sintaxis por lo que no existe retrocompatibilidad. Hay que elegir. En la actualidad la mayoría de librerías son compatibles con ambas versiones, pero cada vez

son más las que al actualizarse dejan de lado la versión antigua en algunas de sus funcionalidades. Dado que además la versión 3 se plantea como un lenguaje perfectamente sólido, parece razonable inclinarse por su uso. A continuación se comentará y justificará brevemente cuáles son las librerías punteras del lenguaje que han sido clave para el desarrollo de este proyecto.

- **Manejo de grandes cantidades de datos**: En el proyecto será indispensable el manejo de grandes cantidades de datos. Esto implicia que se necesitan funcionalidades para leer, escribir y modificar cantidades ingentes de información en estructuras de datos apropiadas. En este sentido, existe *Pandas*, una librería para organizar y analizar datos [08b]. *Pandas* no tiene rival en este aspecto y de hecho *Python* le debe una importante parte de su popularidad. Esta librería facilita muchísimo el trabajo con grandes datasets, permitiendo incluso conectarse a un servidor *SQL*. *Pandas* ofrece una interfaz de alto nivel al usuario pero por debajo utiliza otra librería de más bajo nivel llamada *Numpy* específica para trabajar con *arrays*. En contadas ocasiones se tendrá que utilizar directamente esta última.

- **Visualizaciones**: Cuando se hace minería de datos y finalmente se encuentra una característica de la información relevante hay que poder contar con librerías que ayuden a hacer efectiva su visualización. En esta memoria será recurrente la necesidad de presentar datos para poder extraer conclusiones a partir de las cuáles tomar decisiones debidamente justificadas. *Python* cuenta con una potente librería para esto denominada *Matplotlib* [02]. No obstante, a la hora de hacer visualizaciones relativamente complejas resulta algo tedioso lidiar con ella debido a que es de bajo nivel. Por eso, en muchas ocasiones se utilizará *Seaborn*, que ofrece una interfaz de más alto nivel que será clave en algunas visualizaciones como distribuciones uni/bivariadas, mapas de correlaciones etc.

- **Aprendizaje Automático**: A lo largo de este proyecto será necesario construir clasificadores y regresores utilizando algoritmos pertenecientes al estado del arte del *Machine Learning*. No se van a dedicar recursos a desarrollar algoritmos nuevos por lo que se buscarán opciones ya implementadas. Al ser este un tema muy recurrente en el lenguaje *Python*, son varias las opciones que existen. Una manera apropiada de elegir cuál de ellas utilizar es analizar cuáles son las comunidades más activas de cada una. Un ecosistema vivo de desarrolladores implica ventajas importantes; por un lado, es señal de que se está trabajando para que esté actualizada y refleje los últimos avanzes en investigación y por otro, es más probable encontrar solución a los problemas que vayan surgiendo de su utilización en foros como *Stack Overflow*. Recientemente *KDnuggets* hizo este análisis [15b] y sus resultados se pueden observar en la figura 4.3.

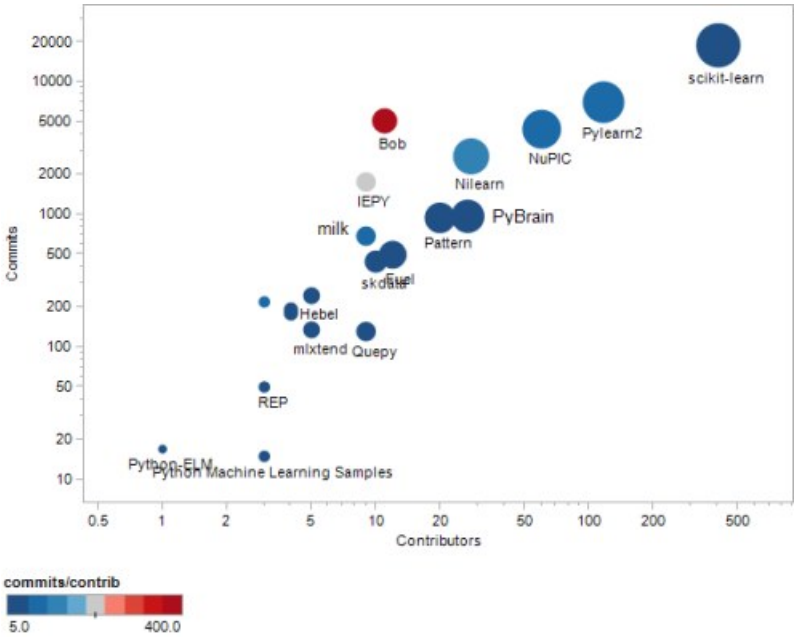


Figura 4.3: Top proyectos *open-source* de Aprendizaje Automático para *Python*. Fuente: *KDnuggets*[15b]

Se ha explorado cada una de ellas y se ha concluído que la más apropiada es *Scikit-learn* debido a que:

1. Es con diferencia la que tiene una comunidad más grande y activa
2. Tiene el abanico más grande de algoritmos implementados
3. Está muy bien integrada con otras librerías de alto nivel como *Pandas*, algo que resultará extremadamente útil.

Como desventaja, se puede decir que no tiene implementados los últimos avances y descubrimientos en *Deep Learning*, por lo que la red de neuronas más compleja que se encontrará es el Perceptrón Multicapa, nada de redes de neuronas convolucionales, recurrentes o profundas [Ben09]. Si se quisiesen hacer pruebas con este tipo de tecnología sería necesarío hacer uso de otras librerías más específicas.

- **Web scrapping**: Como se justificará a lo largo del proyecto, en determinadas ocasiones será necesario descargar recursos almacenados en páginas webs concretas. A menudo, la descarga de manual de estos datos implicaría gastar demasiado tiempo. *Web scrapping* son un conjunto de técnicas que se centran precisamente en automatizar este proceso, que se puede dividir por un lado en la obtención de los recursos web y por otro el procesado de los mismos:
  - **Obtención**: consiste fundamentalmente en descargar una página web completa (en formato `html`). Existen excelentes librerías en *Python* para mandar peticiones a partir de

una `url` y guardar los resultados, como *urllib*[2] o *requests*[3] que implementan módulos para el manejo de `urls` y la comunicación mediante protocolos web como *http* o *https*. No obstante, existen sitios web a los que no les gusta ser el objetivo de *web scrappers*, ya sea por proteger su información o por prevenir sobrecargas de sus servidores. Es fácil para los mismos detectar cuándo una petición proviene de una de las librerías mencionadas de cuando es de un navegador. No sólo eso sino que además se añade otra complicación más: suponiendo que se superase esta barrera, frecuentemente ocurre que el fichero `html` descargado no coincide con el que se descarga si se mandase la petición desde un navegador tradicional. El motivo por el cuál esto ocurre es que al lanzar la petición a partir de la url se descargará el `.html` en su estado inicial, y en ocasiones para que el contenido deseado aparezca primero se tienen que ejecutar algunos fragmentos de código *Javascript* incrustados en la propia página. Estos *scripts* se ejecutan de manera transparente al usuario cuando éste visita la web a través de un navegador, y no simplemente haciendo peticiones *http* desde un script de *Python*.

La idea subyacente es, por tanto, que el *scrapper* interactúe con los sitios web de la misma manera que lo haría un navegador pero en lugar de mostrar la información descargada a través de una interfaz gráfica simplemente la almacene en memoria/disco duro. Afortunadamente, en *Python* existen funcionalidades implementadas para prácticamente cualquier cosa y las librerías que simulan la navegación web no iban a ser la excepción. Hay multitud de opciones, siendo *Scrapy* [08c] posiblemente la más popular debido a su facilidad de uso. No obstante, esta librería es de demasiado alto nivel y ciertos aspectos no son configurables, por lo que se descarta su uso. Además, todavía está en proceso de ser portable a *Python 3*, que es la versión que se ha decidido utilizar. Por otro lado, *Selenium* [04] se muestra como una opción bastante recomendable a la hora de automatizar la navegación web. Cabe destacar que el principal motivo por el que existe *Selenium* no es hacer *web-scraping* sino automatizar la fase de pruebas de aplicaciones web. No obstante, la herrammienta sirve para mucho más que eso. En esta línea, resulta interesante citar la primera frase de la página principal de esta librería:

> 'Selenium automates browsers. That's it! What you do with that power is entirely up to you. Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that.'

Selenium permite, entre otras cosas, intreactuar con las páginas web ejecutando el código *Javascript* de las mismas sin mucha dificultad. A lo largo de este trabajo no será necesario llevar a cabo interacciones complejas, simplemente contar con la funcionalidad de esperar a que ciertos elementos de la página aparezcan como resultado de las pertinentes peticiones internas que se tuviesen que ejecutar antes de volcar la información en una variable.

– **Procesado**: Una vez que se han obtenido los datos deseados en formato `html`, habrá que extraer las partes de información que realmente se necesitan. Por ejemplo, si los datos a obtener eran los que tenía una columna determinada de una página web, una vez que se descarga el fichero habrá que buscar entre todo el código de la página (que puede ser *Javascript*, `html`, *css* etc.) descargada los valores de dicha columna y desechar el resto.

[2] `docs.python.org/3/library/urllib.html`
[3] `docs.python-requests.org/en/master/`

> 'You didn't write that awful page. You're just trying to get some data out of it. Beautiful Soup is here to help'

Son las primeras líneas de la documentación de *BeautifulSoup* [Cru12], una magnífica librería cuyo cometido no es ni más ni menos extraer datos a partir de ficheros `html`. Por supuesto, cada página web estructura la información de manera distinta, por lo que habrá que adecuar el *scrapper* según el caso. *BeautifulSoup* se presenta con métodos que hacen que este trabajo resulte mucho más sencillo. Su metodología se puede resumir en tres sencillos pasos (se parte de una página web que se ha descargado y está almacenada en local):

1. Inspeccionar los elementos de la página web utilizando un navegador como *FireFox* o *Chrome*. La idea es identificar a nivel de código en qué lugar están ubicados los recursos que se quieren extraer.
2. Parsear la página web.
3. Hacer una búsqueda del elemento localizado en el paso 1 sobre la página parseada en el paso 2.

Por supuesto, la complejidad de los filtros de búsqueda está en la mano del usuario. Así, se pueden hacer búsquedas tanto por tipo de elemento `html` como por los atributos de los mismos. También se pueden construir filtros que devuelvan varios resultados. Las opciones son muchas pero la idea siempre es la misma: encontrar los patrones que definen la estructura de la información que se quiere conseguir para luego utilizar los métodos de búsqueda que se adecúen a ellos.

En este aspecto, *BeautifulSoup* no tiene rival, es con diferencia la librería para parsear y analizar código `html` más completa y con una comunidad más activa de *Python*, por lo que sin duda alguna será la elegida para este proyecto.

4.2. Marco Regulador

Para asegurar que la realización del proyecto no conlleve implicaciones legales graves, hay ciertas cosas que hay que tener en cuenta antes de utilizar los *web-scrappers*: la legalidad vigente en el país origen, las condiciones legales del sitio web, derechos de autor, de marca registrada etc. Las fuentes que se van a scrappear **son de acceso público** y el uso que se va a hacer de los mismos no lo es, por lo que en términos generales será tan legal como pueda serlo acceder a sus datos con un navegador web. Por otro lado, dado que las fuentes de extracción son estadounidenses, aplican las siguientes consideraciones:

1. Siempre y cuando no '*crawleen*' a tasas disruptivas, los *scrappers* no incumplen ningún contrato de términos de uso ni suponen el cometimiento de un crimen. Por ello, y por ética se llevará un control exhaustivo de las peticiones que se envíen a las fuentes. Un abuso de este tipo podría ocasionar fallos en los servidores de las fuentes.

2. Los acuerdos de usuario de las páginas web no son exigibles por ley[4] como acuerdos de navegación válidos porque las empresas no los hacen suficientemente visibles para los visitantes.

[4] `http://www.nelsonmullins.com/articles/browse-wrap`

3. Los *scrappers* acceden a las páginas web como visitantes, siguiendo un camino similar al de los motores de búsqueda. Esto se puede efectuar sin problemas sin tener que registrarse como usuario (aceptando explícitamente los términos).

4. Existen casos jurídicos[5] donde las cortes decretaron que la simple colocación de un enlace a los términos de uso en la parte más inferior de la página no es suficiente para 'dar lugar a una notificación constructiva'. En otras palabras, no hay nada en una página pública que simplemente por el hecho de ser accedida esté sujeto a términos contractuales. Los *web scrappers* que se implementarán en este proyecto no hacen ningún acuerdo explícito o implícito, por lo que no violan ningún contrato.

5. Es importante recalcar también el uso de los datos personales: En el caso de las redes sociales, tanto en Estados Unidos como España se asigna **exclusivamente** a los visitantes que se convierten en usuarios las habilidades de libre acceso a perfiles completos de otros usuarios, la búsqueda e identificación de amigos o conexiones en común y poder contactar con usuarios directamente. Los *web scrappers* que se implementen en este proyecto no llevarán a cabo ninguna de estas acciones así que no estarán haciendo uso de un 'acceso no autorizado' y no violarán el CFAA[6].

El *web scrapping* es la única parte del proyecto que necesitaba ser colocada bajo el marco regulador explícitamente. La naturaleza del resto de operaciones no merece aclaraciones legales, ya que todos los datos de empresas que se manipularán (pertenecientes a Global Incubator) han sido obtenidos de manera legal y se comercializan de acuerdo con las disposiciones de la Ley 15/1999, de 13 de Diciembre, de Protección de Datos de Carácter Personal (LOPD).

4.3. Equipo y modus operandi

Estas son algunas consideraciones relevantes acerca de las herramientas utilizadas en el modus operandi:

- **Windows**: Para la realización de la mayoría de las partes del proyecto se han utilizado dos portátiles. Uno utilizado en el lugar de trabajo (oficina) con *Windows 10* y otro fuera del lugar del trabajo con *Windows 8.1*.

- **Debian**: No obstante, existirán muchas tareas que, al manejar grandes cantidades de datos, no pueden llevarse a cabo en un ordenador de uso habitual. No sólo es inviable el entrenamiento de ciertos modelos de Aprendizaje Automático sino que simplemente mapear una función no demasiado compleja sobre una columna de más de un millón de celdas puede ser extremadamente costoso. Para poder ejecutar todas las operaciones requeridas se hará uso de un servidor dedicado alquilado por Global Incubator que cuenta con la distribución *Debian* basada en *Unix*, 40 procesadores, 256gb de memoria RAM y 4tb de disco duro. La captura correspondiente a la figura 4.4 muestra la ejecución del comando `htop`[7] durante la ejecución de una tarea de uso intensivo en CPU. En general, siempre que se puedan paralelizar los procesos así se hará. En ocasiones la paralelización será muy sencilla, especialmente al utilizar algunos

[5] `http://newmedialaw.proskauer.com/2014/09/08/browsewrap-agreement-held-unenforceable-against-consumer-due-to-insufficient-notice/`

[6] `https://en.wikipedia.org/wiki/Computer_Fraud_and_Abuse_Act`

[7] Comando de *Linux* que entre otras cosas, muestra el uso actual de los recursos de la máquina

algoritmos de Aprendizaje Automático de *Scikit-learn* ya que las propias funciones vienen con un parámetro *n_jobs* que indica el número de procesadores a utilizar. Por ejemplo, como se verá en la sección 7.1, el entrenamiento de 500 árboles de decisión de un *Random Forest* resultamente fácilmente paralelizable de esta manera. No obstante, habrá casos (como en la extracción de *tags* de la sección 5.2), donde sea necesario mapear funciones sobre una gran cantidad de datos y habrá que lanzar los múltiples procesos explícitamente. Afortunadamente, la mayoría de estas situaciones se corresponderán con lo que se conoce como *Embarrassingly parallel problems*, problemas donde dividir la tarea principal en múltiples subtareas es sencillo y no es necesario abordar problemas típicos de la paralelización como las condiciones de carrera o memoria compartida. Para llevarlas a cabo merece la pena recalcar que se paralelizarán procesos y no hilos ya que el intérprete de *Python* no permite ejecutar los hilos de un mismo proceso en paralelo debido al proceso interno *Global Interpreter Lock* (GIL)[Bea10]

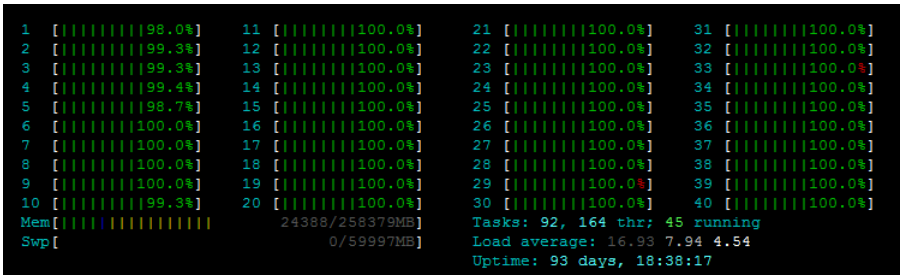


Figura 4.4: Uso de los recursos del servidor realizando una tarea pesada

- **Putty**: Normalmente, se diseñará y ejecutará a pequeña escala el código en local para posteriormente ser ejecutado en la nube sobre todo el conjunto del problema. Se utilizará el protocolo SSH[8] para conectarse al servidor desde local mediante el programa *Putty* debidamente configurado con una clave RSA[9]. De está manera se tendrá pleno acceso al servidor desde local de manera segura.

- **WinSCP**: Para poder ejecutar los programas en el servidor no basta con tener acceso al mismo sino que será necesario transmitir los ficheros necesarios (código y datos) de alguna manera. Para ello se utilizará el protocolo SFTP[10] perfectamente implementado en el programa *WinSCP*.

- **SQL**: Dado que se manejarán una gran cantidad de datos es muy recomendable contar con un sistema gestor de los mismos. Se ha elegido SQL[11] y concretamente MySQL[12] dado que es el estándar de la industria y es perfectamente compatible con la librería *Pandas*, de tal

[8]Protocolo utilizado para acceder de manera segura a máquinas remotas sobre un red no segura.
[9]Criptosistema ampliamente utilizado en transimisiones de datos seguras pionero en el uso del modelo de llave pública para encriptar y llave privada para desencriptar.
[10]Protocolo de red que permite la transmisión segura de ficheros extendiendo a SSH.
[11]Lenguage de programación especialmente diseñado para la gestión de datos almacenados en una base de datos relacional.
[12]Sistema gestor de bases de datos relaciones.

manera que se almacenarán los datos en el servidor SQL, se accederá a los mismos a través de *Pandas* y una vez en *Python* se realizarán las operaciones necesarias.

Parte III

Construcción del modelo

Capítulo 5

Determinando la actividad de la empresa

La determinación de la actividad de una empresa se ha elegido como primer paso del método de valoración porque es un factor altamente determinante a partir del cuál se tomarán muchas otras decisiones. Por ejemplo, para estimar el WACC o los ingresos anuales se deberá conocer con anterioridad a qué se dedica la compañía (con la máxima especificidad). La actividad de la empresa viene definida a distintos niveles: el más amplio sería el sector al que pertenece, bajando un nivel estaría su industria y un escalón más abajo estarían las palabras clave que dan una imagen más específica, ya sea porque definen el tipo de tecnología utilizada o porque concretan una o varias sub-industrias.

Para poder efectuar esta determinación inicialmente se abordó el problema con técnicas de Aprendizaje Automático (sección 5.1) pero los resultados no fueron satisfactorios por lo que se planteó otra solución basada en un sistema experto de reglas (sección 5.2).

5.1. Mediante Aprendizaje Automático

5.1.1. Entendimiento del negocio

Cuando una compañía es muy joven, son varias las decisiones que el CEO[1] debe tomar activamente para sacar adelante su startup. Estos factores que pueden ser modificados directamente van desde definir una visión clara que sea compartida por el público objetivo hasta tener la capacidad de adaptarse correctamente a los cambios producidos en las primeras iteraciones del desarrollo del producto mínimo viable.

No obstante, uno de los mayores determinantes del éxito de una startup -y por tanto, de su valoración- son las fuerzas del mercado propias del sector y la industria en los que se sitúa. Desafortunadamente para un CEO, estos son factores en los que es muy difícil influir. Estas son algunas de las características de sectores e industrias que influyen directamente en la disposición de los inversores a llevar a cabo una financiación:

[1] Chief Executive Officer o Director Ejecutivo es aquella persona encargada de máxima autoridad de la llamada gestión y dirección administrativa en una organización o institución

- El equilibrio o desequilibrio entre la demanda y la oferta de dinero.
- La frecuencia y tamaño de las últimas salidas producidas, ya sea por venta a terceros, recompra por parte de los emprendedores o salida a bolsa.
- El estado de madurez del mercado.
- El porcentaje de startups que fracasan.
- El Coste Medio Ponderado de capital ($WACC$) medio.

Resulta evidente que los factores listados son interdependientes. De hecho, esclarecer de manera objetiva cuál de ellos debería ser el principal en un método de valoración carecería de sentido, al menos a priori. En cualquier caso, será beneficioso conocer a qué sector e industrias pertenece una startup, ya que como se puede ver, dicha información tiene implicaciones directas sobre la frecuencia y el volumen de las inversiones que una compañía joven puede recibir, y por lo tanto sobre el éxito ó fracaso que se le puede diagnosticar.

La razón y objetivo principal de ésta sección es generar toda información relacionada con el sector y/o industria/s de una compañía considerada como input relevante para posteriormente calcular su valoración. En esto se incluyen el nombre del sector y la industria, pero en última instancia lo más importante es identificar de alguna manera a qué se dedica la empresa, por lo que recuperar cualquier palabra clave será útil. Si es algo genérica, vale, y si también se extrae algo específico, pues mejor. Poniendo todo en contexto, cuanta más información se pueda extraer de las empresas incluidas en la base de datos los resultados obtenidos serán potencialmente mejores. Es importante que el cliente perciba que la aplicación contiene información de calidad.

Una vez extraídos el sector y la industria en los que opera la empresa, se combinarán ambos con ciertos conocimientos obtenidos del exterior (artículos y estudios estudios estadísticos) para generar nuevos atributos potencialmente relevantes. Por ejemplo, se buscará un listado de tasas de quiebra y $WACCs$ medios por sector/industria para posteriormente atribuírselos a cada startup en función del tipo de actividad que desarrolle (sección 5.3). Por supuesto, ésta asignación no supondrá que esos sean efectivamente los valores para esa startup en concreto, simplemente se le atribuirán porque lo que sí que se estará asumiendo es que deben influir en su valoración.

5.1.2. Entendiendo los datos

Dado que se parte de una base de conocimiento con bastante información, parece una buena idea utilizarla para crear (entrenar) un modelo que sea capaz de generalizar los conocimientos y patrones implícitos en ella. Con dicho modelo se espera inferir cuál es el sector/industria de una startup analizando el texto previamente extraído de su página web si esto es lo único que se conoce de ella, o de su descripción si ésta está incluida en el estado actual de la base de datos.

Metadatos : En concreto, se dispone de 1.092.849[2] startups, 249.493 de las cuáles tienen asignada una o más industrias, es decir el 23 %: Como se explicará más adelante, el atributo que se utilizará para la clasificación es la descripción; por tanto,se utilizará el subconjunto coloreado de azul para entrenar el modelo y para después aplicarlo sobre el subconjunto coloreado de verde (*target subset*).

[2]Global Incubator dispone de un número mayor (+1,5M) y con menos valores nulos que centraliza a partir de varias fuentes, pero este fue el dataset utilizado por el autor debido a que en ese momento el resto de los datos estaba en proceso de unificación y homogeneización.

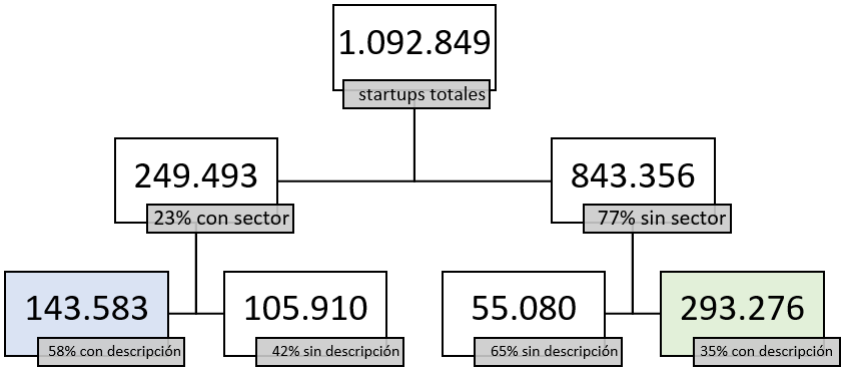


Figura 5.1: Metadatos del dataset relacionados con la industria y descripción

De esto se desprende que en el mejor de los casos se conseguiría pasar de tener un 23 % a un 50 % de startups con al menos un sector asignado. No obstante, se recuerda que el objetivo principal de este dataset es construir un modelo predictivo. El sistema final recopilará datos de distintas fuentes a partir de los cuáles generará conocimiento utilizando lo aprendido en ésta sección.

Se dispone de 110 industrias diferentes. En la figura 5.2 se muestra claramente que hay algunas mucho más frecuentes que otras en el dataset. Como hay 110 distintas, se han omitido del histograma los nombres de cada una de ellas. Se tiene que por ejemplo, las más comunes son *e-commerce*, *mobile*, *healthcare*, *education* y *marketing*; mientras que las menos frecuentes son *professional services*, *event* y *android*.

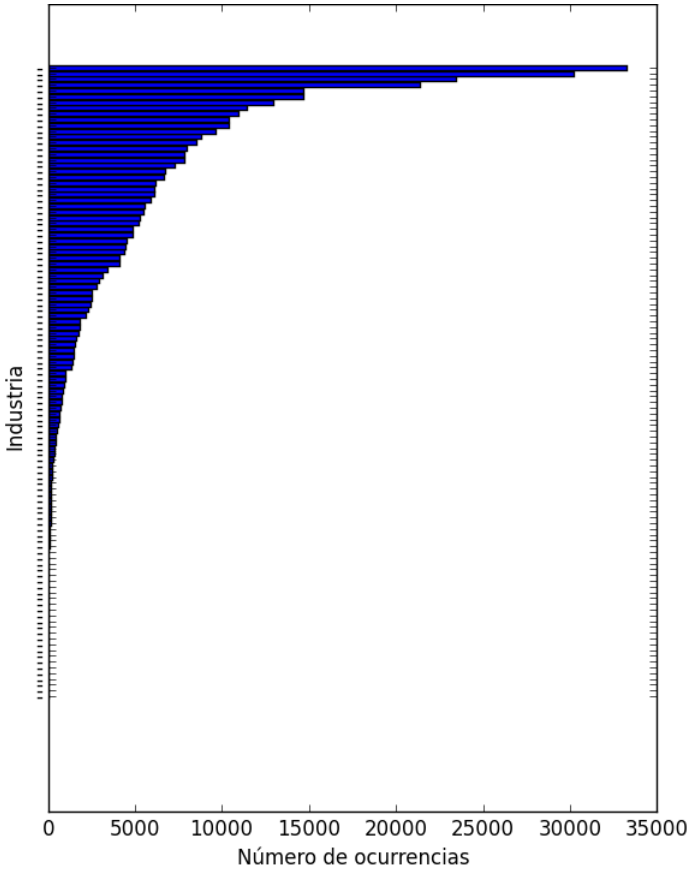


Figura 5.2: Frecuencias de industrias en el dataset

De dicha lista se desprende que, como era de esperar, existen relaciones jerárquicas entre algunos sectores (*android* es un subgrupo de *mobile* de la misma manera que *bitcoin* lo es de *finance*). No obstante, las relaciones no son estrictamente jerárquicas, ya que por ejemplo existen palabras clave (*analytics*, *machine learning*) que de alguna manera definen la actividad de la empresa y están incluidas en el campo objetivo como si fuesen industrias en sí pero resulta difícil asignarlas un sector concreto ya que más que industrias, definen un conjunto de tecnologías. Este tipo de relaciones entre las posibles categorías son especialmente difíciles de detectar por un algoritmo de Aprendizaje Automático y en definitiva dependerá de la calidad de los ejemplos del entrenamiento.

Se van a utilizar los 249.493 ejemplos etiquetados para realizar un aprendizaje supervisado con el objetivo de aplicar posteriormente lo aprendido para predecir las industrias a las que pertenecen las startups no etiquetadas. La cuestión es entonces discernir si una startup pertenece o no a una industria, así que se trata de un problema de clasificación. No sólo eso, sino que además una startup podrá pertenecer a ninguna, una o varias industrias, por lo que se dice que se trata de una clasificación multi-etiqueta. Otra manera de ver el problema tras ésta última apreciación es como 110 problemas de clasificación binaria independientes. Donde para cada uno de ellos se decide si la

startup pertenece o no a una industria en concreto.

Puede parecer a priori que el hecho de que exista un desequilibrio importante **entre las distintas industrias** no supondrá un problema ya que se utilizarán clasificadores independientes para cada una de ellas. No obstante, el hecho de **para una clase concreta** más del 95 % de instancias estén etiquetadas negativamente sí que podría ser un problema, ya que probablemente exista cierta tendencia en el clasificador a decidirse atendiendo a cuál es la opción más frecuente independientemente de la descripción en sí. En especial, los modelos lineales suelen ser más sensibles a este desequilibrio y las predicciones finales estarán muy influenciadas por la función de pérdida que se intente minimizar.

En este caso, se intuye claramente que el atributo principal que se puede utilizar para llevar a cabo la clasificación de las 110 distintas industrias es la descripción.

5.1.3. Preparación de los datos

Representación del texto : Como se explicó anteriormente es necesario transformar el texto crudo en vectores numéricos que un algoritmo pueda entender. Se elegirá la transformación *TF-IDF* (explicada en la subsección 3.2) por ser bastante más completa y robusta que el conteo simple de palabras. Estas son algunas consideraciones notables respecto a esta decisión:

- *S*e dispone de textos de muy variada longitud. Lo que se puede probar con unas pocas líneas:

Código 5.1: Estadísticos del número de palabras de las descripciones

```
#Cargar en memoria la columna que contiene las descripciones
df=pd.read_csv(path)
#Descartamos los valores nulos
df.dropna(inplace=True)
#Generamos nueva columna con el numero de palabras
df['n_words']=df['Description'].map(lambda d: len(d.split()))
#Mostramos indicadores
print(df['n_words'].describe())
```

Lo que imprime por consola:

```
count    436859.000000
mean         20.133196
std          22.010188
min           1.000000
25%          11.000000
50%          18.000000
75%          24.000000
max        8983.000000
Name: n_words, dtype: float64
```

La alta desviación típica sugiere no sólo que el conteo de palabras no sería efectivo, sino que la frecuencia de cada término deberá ser tenida en cuenta en relación a la descripción y en relación a la colección de descripciones.

- El hecho de que una compañía pertenezca a un sector u otro va a venir más determinado por la presencia esporádica de palabras clave concretas más que por la presencia y distribución de conjuntos de palabras. Se necesita una transformación que potencie este tipo de ocurrencias y esa es una de las principales ventajas del *TF-IDF*.
- La principal limitación del *TF-IDF* es que al estar basado en el modelo de Bolsa de Palabras, no se tiene en cuenta ni la posición relativa de los términos ni su información semántica (sólo es útil a nivel léxico). No obstante, dadas las características del problema, es un precio que merece la pena pagar.

En aprendizaje supervisado, suele ser buena práctica hacer un análisis previo de qué atributos realmente aportan valor y cuáles se limitan a introducir ruido. Existen multitud de técnicas ya implementadas que sirven para realizar este tipo de análisis [Sci14a]. La decisión de llevarlos o no a cabo depende también del tipo de técnicas que se vayan a utilizar posteriormente para entrenar el modelo, ya que existen algoritmos más robustos al ruido que otros. En este caso los atributos serán las palabras pertenecientes al diccionario generado. En las pruebas a realizar se contemplarán los siguientes hiperparámetros en lo que a transformación de los datos y selección de atributos (construcción del diccionario/vocabulario) se refiere:

- *Stop words*: Son conjuntos predeterminados de palabras que se ignorarán en la construcción del diccionario. Es útil cuando se presupone que existen palabras que no aportan absolutamente nada de información, como artículos, preposiciones etc. Se probará a utilizar las *stop words* del idioma inglés (definidas por *Scikit-learn*), ya que en ese idioma están las descripciones de la gran mayoría de empresas del dataset.
- *Frecuencia mínima*: Se puede establecer un umbral que actúa como la frecuencia mínima de una palabra para que esta sea incluída en el diccionario o vocabulario.
- *Rango de n-gramas*: Si se configura con el valor *(1,2)* se utilizarán como atributos tanto las palabras sueltas como pares de palabras (que vayan seguidas en el texto). Así, '*machine learning*' o 'health insurance' podrían ser considerados como atributos. Favorece la detección de relaciones de cierta complejidad pero aumenta mucho el número de atributos, por lo que debe utilizarse con cuidado.
- *Usar* ***IDF***: Se puede elegir si utilizar o no el segundo término del *TF-IDF*, es decir, la ecuación 3.2.
- *Aplicar escalado sublineal a* ***TF***: Aplicar o no la transformación $1 + \log TF$ al término *TF* correspondiente a la ecuación 3.1

5.1.4. Modelado

Los modelos/estimadores que se probarán serán los que se listaron en la sección 3.2. Dado que en dicha colección de algoritmos hay ejemplos de familias muy diversas, se espera poder explorar todo el espacio del problema para que posteriormente en la fase de evaluación se puedan sacar conclusiones firmes. Además, se probarán distintas configuraciones para cada uno de ellos. En este sentido se hará uso de la fantástica herramienta implementada en *Scikit-learn* llamada *Grid-search* [Sci14b] mediante la cuál se hace una exploración del espacio de configuraciones/hiperparámetros posibles del estimador en su totalidad (clasificador + transformador de las variables) y se selecciona aquella

que dé mejores resultados. Las configuraciones a probar obviamente variarán de un algoritmo a otro. Así, para las máquinas de soporte vectorial se probarán distintos *kernels*, se jugará con α en *Lasso* y se observará el comportamiento de distintas funciones de pérdida para aquellos clasificadores que las soporten.

5.1.5. Evaluación

A continuación se ejecutarán los experimentos a partir de las decisiones que se tomaron en la fase de Modelado. No obstante, hay una importante decisión que tomar relativa a la ejecución del *grid search*. Hay que explicitar cuál será la función que el algoritmo intente maximizar, es decir, bajo qué criterios durante la ejecución se decidirá que una configuración de parámetros es mejor que otra.

La implementación de *grid-search* por parte de *Scikit-learn* está preparada para que el usuario pueda definir sus propias funciones que luego internamente se buscará maximizar. No obstante, en este caso podemos utilizar algunas de las que ya están definidas. Concretamente, se podría elegir entre la precisión, exhaustividad y *F1 score* (listadas en la subsección 3.4.1).

Siguiendo con la metodología *CRISP-DM* explicada en la sección 3.6, para tomar esta decisión adecuadamente es un requisito indispensable recordar los objetivos del problema, por lo que se debe reconectar con la fase de 'Entendimiento del negocio' (sección 5.1.1 en este mismo capítulo). Una vez repasados el problema que se quiere resolver y los objetivos del mismo se decide que la función a maximizar durante la búsqueda *grid-search* sea la exhaustividad (*recall*): El principal objetivo es extraer información que sea relevante en cuanto a la actividad de la empresa, cuanta más mejor. Si se tiene en cuenta que además una empresa perfectamente puede tener 5 o 6 etiquetas a la vez se llega a la conclusión de que en este problema es mejor tener *algunos* falsos positivos pero *muchos* verdaderos positivos, que es lo que ocurre al priorizar la exhaustividad, que tener *pocos* falsos positivos pero *pocos* verdaderos positivos, que es lo ocurre al priorizar la precisión. El término medio sería intentar maximizar el *F1-score*, pero al ser los objetivos del problema como se ha explicado, se decide que será más apropiado garantizar una buena exhaustividad.

La función *GridSearchCV()*[Sci14b] será entonces una pieza clave en la prueba. Cuenta con parámetros que merece la pena pararse a explicar:

- **score**: indica la función a maximizar. En este caso será *'recall'*.
- **n_jobs**: indica el número de procesadores a utilizar, con el valor -1 utiliza todos que en este caso son 40. Se dedicarán todos los recursos de la máquina a esta tarea ya que son muchas las combinaciones posibles de configuraciones.
- **refit**: si se establece como *True*, el método devolverá la instancia del estimador **entrenado** con los parámetros óptimos. Esto resultará útil ya que se deseará mediar otras métricas y probar a hacer unas cuantas predicciones sobre los datos para tener una mejor idea sobre la capacidad generalizadora del modelo 'óptimo'.
- **cv**: entero que sirve para indicar el tamaño o número de *folds* de la validación cruzada. Valores altos hacen que la evaluación sea muy precisa, pero implica más consumo de recursos. El valor por defecto 3 es adecuado para la prueba.

En la tabla 5.1 se presentan los resultados de la prueba. Aparte de las métricas mencionadas, se indica el tiempo que se ha tardado en realizar la prueba y las descripciones que han quedado sin

industria/sector. Esta última métrica será útil para hacerse una idea la completitud de la solución. Por motivos estéticos, se toman las siguientes abreviaturas: P = precisión (%), R = exhaustividad (%), T = tiempo (segundos) y E = ejemplos sin industria predicha (%). De estas pruebas se concluye que:

- Existe prácticamente unanimidad en la configuración óptima del transformador *TF-IDF*.
  - La frecuencia mínima de una palabra para sea incluida en el diccionario debe ser 1e-07 (se probó 1e-06, 1e-07 y 1e-08).
  - Merece la pena utilizar bigramas aparte de las palabras sueltas ya que enriquece de manera considerada el vocabulario el diccionario.
  - Es recomendable utilizar el término *IDF* (ecuación 3.2) que ajusta la importancia de las palabras teniendo en cuenta su frecuencia en la totalidad de los documentos.
  - Aplicar la transformación $1 + \log TF$ al término *TF* (ecuación 3.1) da mejores resultados que no hacerlo.
- Se observa muy bien el *trade-off* existente entre la precisión y la exhaustividad. El clasificador que mejor ejemplifica este efecto es el *SGD* (que hace uso del *Stochastic Gradient Descent*[Bot10] para minimizar la función de pérdida y el regularizador), con un 78 % de precisión pero con una exhaustividad de sólo un 17 %. Este resultado provoca una tendencia del estimador resultante hacia posiciones conservadoras y desconfiadas, habiendo preferido 'no mojarse' en el 73 % de los casos, motivando su baja exhaustividad.
- Teniendo en cuenta los objetivos del problema, el clasificador que mejor resultados ofrece es la máquina de soporte vectorial con *kernel* lineal *LinearSVC*. En Aprendizaje Automático, más complejo/sofisticado no implica mejores resultados como se puede ver en este caso donde la sencillez y elegancia de una máquina de soporte vectorial con *kernel* lineal a desbancado a sofisticados algoritmos como *RandomForests* y *ExtraTrees*.
- Aún en el caso del *LinearSVC*, y a pesar de que se reconozca que dar falsos negativos no es tan grave, hay demasiados y los resultados no son suficientemente buenos. Por lo tanto el modelo no es suficientemente satisfactorio.

Tras haber probado tantas combinaciones de configuraciones en el clasificador y transformador, merece la pena comprobar si los 'insatisfactorios' resultados están motivados por un factor externo al tipo de modelo.

Identificando el problema

Analizando las descripciones de startups etiquetadas con industrias se observa una característica problemática: hay demasiados *outliers*. En la figura 5.3 se han seleccionado 4 ejemplos tipo de los que se desprenden algunas conclusiones, algunas de ellas sirviendo para explicar las hipótesis acerca de los mediocres resultados obtenidos, y otras para dar ideas de cómo solucionar el problema:

Resultados de Grid Search sobre parámetros de clasificadores							
Clasificador	Mejores parámetros clasificador	Mejores parámetros TF-IDF	T	P	R	F1	E
Ridge	α=0.05 solver='auto'	ngram=(1,2) min_df=1e-07 use_idf=True sublinear_tf=True	5409	40	25	29	51
Perceptron	α=0.05 penalty='elasticnet'	ngram=(1,2) min_df=1e-07 use_idf=True sublinear_tf=True	6505	33	28	29	32
Passive Agressive	C=0.5 loss='hinge'	ngram=(1,2) min_df=1e-07 use_idf=True sublinear_tf=True	454	42	29	32	46
KNeighbors	metric='minkowski' weights='distance'	ngram=(1,1) min_df=1e-07 use_idf=False sublinear_tf=False	2344	67	13	22	72
Random Forests	criterion='gini' max_depth=None	ngram=(1,2) min_df=1e-07 use_idf=True sublinear_tf=True	4120	65	27	36	55
Extra Trees	criterion='gini' max_depth=None	ngram=(1,2) min_df=1e-07 use_idf=True sublinear_tf=True	4893	62	29	39	49
LinearSVC	multi_class= 'crammer_single' loss='squared_hinge' penaly='l2'	ngram=(1,2) min_df=1e-07 use_idf=True sublinear_tf=True	7566	68	44	54	28
SGD	α=0.001 average=1 warm_start=False	ngram=(1,2) min_df=1e-07 use_idf=False sublinear_tf=True	1587	78	17	26	73
Multinomial NB	α=05 fit_prior=False	ngram=(1,2) min_df=1e-07 use_idf=False sublinear_tf=True	330	64	22	32	62
Bernoulli NB	α=05 fit_prior=False	ngram=(1,1) min_df=1e-07 use_idf=False sublinear_tf=True	430	47	7	12	84

Cuadro 5.1: Resultados del la prueba de clasificadores de textos

```
1, id: 293044
      Description:
Fenway Summer is a consumer finance advisory and investment firm
headquartered in the historic Georgetown neighborhood of Washington,
DC.
      Real industries: ['finance']
2, id: 589533
      Description:
Seventh Continent, a business game platform, enables the exchange of
digital commodities created and traded in bitcoin by virtual
companies.
      Real industries: ['e-commerce', 'bitcoin', 'gaming']
3, id: 349539
      Description:
Design your own fashion design fashion
      Real industries: ['e-commerce']
4, id: 672569
      Description:
Generate more mobile leads and sales with Viva Tags. Our mobile
commerce platform simplifies mobile marketing, purchases, and
payments. Alcohol & payments.
      Real industries: ['finance', 'food & beverages', 'mobile',
'payments', 'transportation']
```

Figura 5.3: Ejemplos mal etiquetados del dataset de entrenamiento

- Hay muchas descripciones que manifiestan que claramente la empresa pertenece a determinadas industrias pero no están etiquetadas consecuentemente. Así, se tiene que:

  1. La empresa 1 debería estar etiquetada como empresa de asesoría o consultoría pero no es así. También debería tener la etiqueta *consumers*.
  2. La empresa 3 debería estar etiquetada como *fashion* y *design* pero no lo está.
  3. La empresa 4 debería estar etiqeutada como empresa de *marketing* y *e-commerce* pero no es el caso.

  Con un dataset lleno de ejemplos mal etiquetados es difícil entrenar un modelo y obtener buenos resultados. Esta 'calidad' a la que se hace referencia es difícilmente medible de manera automática al tener una marcada connotación cualitativa. Por lo tanto la manera de estimar la frecuencia de aparición del tipo de ejemplos de la figura 5.3 es revisar manualmente un conjunto aleatorio de descripciones. Se han observado 50 y 27 de ellas no tienen un buen etiquetado (no todos los casos son igual de *malos*). Si todos las descripciones tuviesen los mismos errores de etiquetado, este hecho no tendría por qué justificar una baja precisión o exhaustividad (el modelo podría ser preciso y exhaustivo etiquetando *mal* las descripciones), pero el problema es que en algunos casos la actividad está bien definida y en otros no, y la distribución de los datos resultante confunde fácilmente al clasificador. Cabe destacar que el reconocimiento de este hecho debió haber tenido lugar en la fase de Entendimiento de los

datos (sección 5.1.2). Haberlo detectado en el momento adecuado hubiese ahorrado recursos ya que se hubiese abarcado el problema de otra forma (el correspondiente retraso se ilustra en el diagrama de *Gantt* del apéndice B.1). En cualquier caso, al seguir la metodología *CRISP-DM* y tener que pasar por la fase de evaluación, ha permitido que este error no vaya a más allá de la primera iteración. Hubiese sido mucho más grave detectar esta anomalía en la fase de implementación del sistema.

- Merece la pena hacer la apreciación de que hay casos donde la actividad de la empresa no está reflejada porque el etiquetado a menudo no representa las relaciones jerárquicas de los sectores. Así se tiene por ejemplo la empresa 2, al ser de *bitcoin*, también debería de ser de finanzas, pero no es así.

- Independientemente del mal etiquetado de varias de las empresas, parece que en muchos casos el sector/industria al que pertenece la empresa viene totalmente identificado en forma de palabras clave en su descripción.

Se determina por tanto que el dataset no es apto para un aprendizaje óptimo, por lo que se descarta la construcción de un modelo de *Machine Learning* para determinar la actividad de una empresa a partir de su descripción.

Por otro lado, dado que existen ciertos conjuntos de palabras clave que identifican a cada sector y que aparecen frecuentemente en las correspondientes descripciones se intentará abordar el problema desde una perspectiva de programación más clásica: un sistema experto de reglas.

5.2. Mediante árbol con reglas

En esta sección se va a desarrollar un método para extraer la industria y el sector de una startup a partir de un texto de la misma.

La idea principal consiste en crear una estructura de reglas que comprueben si ciertas palabras están en el texto para decidir si incluir un sector concreto o no. Una de las ventajas principales que se espera obtener es la capacidad de, una vez detectada una industria concreta, asignar también a la startup el sector que la engloba y que no aparece en el texto. Por ello, como se explicará más adelante, su mecanismo será más elaborado y complejo que simplemente incluir *finance* porque *finance* existe en la descripción (por ejemplo).

5.2.1. Redefiniendo el conjunto de industrias y sectores

Dado que en esta ocasión no se depende de ningún dataset (no hay que entrenar un modelo, sino directamente construirlo), se tiene total libertad para definir cuáles serán las industrias y sectores posibles que serán asignadas a cada startup.

Existen varias posibilidades:

- **Standard Industrial Classification, SIC:** Es un sistema que se utiliza para clasificar industrias utilizando un código de 4 dígitos desarrollado por Estados Unidos. Se utiliza mucho. Por ejemplo, suele venir en cualquier base de datos con información cuantitativa de corporaciones y es especialmente útil para cruzar datos de las empresas [37].

- **North American Industry Classification System, NAICS:** Es un sistema de clasificación de industrias que utiliza un código de 6 dígitos. También se utiliza bastante pero son varias las regiones y organismos que todavía prefieren el SIC [97].

- **AngelList:** Es una plataforma comercial estadounidense que donde personas, startups e inversores se dan de alta. Tienen su propia clasificación de industrias [3].

Aunque tanto como *NAICS* y *SIC* ofrecen una taxonomía aceptable, con jerarquías semánticamente completas que van de lo más general a lo más específico, la clasificación que se utilizará será la propuesta por *AngelList*. Las razones son las siguientes:

- **Eficacia**: *NAICS* y *SIC*, pensados para recoger información acerca de la actividad económica de todo tipo de empresas, intentan abarcar todos los sectores posibles con el mismo nivel de detalle, por lo que sus árboles jerárquicos cuentan con varios millones de nodos. Por su parte, *AngelList* apenas tiene registrados 1800. Lo bueno es precisamente que entre esos 1800 se encuentran los sectores e industrias más populares entre startups, con bastante más nivel de detalle que los mencionados estándares. Así, por ejemplo, términos como *Machine Learning*, *Virtual Reality*, *Internet of Things* o *Android* aparecen claramente diferenciados del resto mientras que en *NAICS* y *SIC* ni siquiera se muestran.

- **Eficiencia**: Como se verá posteriormente, se deberá ejecutar una regla como mínimo por cada nodo del árbol. Teniendo en cuenta que la gran mayoría de los varios millones de nodos de las jerarquías de *NAICS* y *SIC* no darán resultados, resulta mucho más eficiente tener condensados en el árbol exclusivamente aquellos términos que efectivamente van a tener más probabilidad de hacer *matching*.

- **Actualizado**: *NAICS* y *SIC* datan de varios años atrás (1937 y 1997, respectivamente) y al ser tan grandes resulta algo complicado reorganizar las ramas del árbol o añadir nuevos nodos según van apareciendo y desapareciendo industrias. Por su parte, *AngelList* responde prácticamente en tiempo real a las tendencias del mercado adaptándose especialmente bien al crecimiento tecnológico y aparición de nuevas industrias de los últimos años, ya que parte de las inversiones recibidas por las *startups* situadas en la vanguardia tecnológica se producen gracias a las conexiones que se dan en este tipo de ecosistemas web.

5.2.2. Análisis y diseño

Como se viene viendo, el principal objetivo es que dado un texto, el sistema sea capaz de extraer las etiquetas que mejor definan la actividad económica de la empresa. Se definen, por tanto, los módulos del problema a abordar a modo de requisitos de sistema en la tabla 5.2.

5.2.3. Implementación

Resulta evidente que la estructura de datos óptima será un árbol. El principal objetivo es construir una representación de la jerarquía de industrias que aparece en `www.angel.co/markets`. Al estar almacenado de dicha manera, será muy sencillo añadir las 'industrias padre' una vez se haga *match* en algún nodo interno del árbol.

[3] `www.angel.co/markets`

Requisitos de sistema del extractor de Sectores e Industrias			
ID	Tipo	Prioridad	Descripción
01	Funcional	Muy alta	El sistema debe ser capaz de detectar sectores (actividades generales de la startup) a partir del texto.
02	Funcional	Muy alta	El sistema debe ser capaz de detectar industrias (actividades específicas de la startup) a partir del texto.
03	Funcional	Muy alta	El sistema debe poder crear la jerarquía a partir del fichero `.html` descargado de `https://angel.co/markets`.
04	Funcional	Muy alta	El sistema debe poder almacenar la jerarquía de industrias y sectores en memoria.
07	Funcional	Media	El sistema debe almacenar todos los metadatos listados individualmente en cada nodo de la jerarquía de *AngelList*.
05	No funcional	Alta	El sistema debe ser capaz de procesar 100000 textos previamente descargados de una longitud media de 1000 caracteres en menos de 5 minutos
06	No funcional	Media	El sistema debe ser capaz de funcionar en entornos Windows
09	No funcional	Muy Alta	El sistema debe ser capaz de funcionar en entornos Linux

Cuadro 5.2: Requisitos de sistema del extractor de Sectores e Industrias

Desafortunadamente, *Python* no tiene la estructura de datos de tipo árbol implementada por defecto. No obstante, dada la versatilidad y popularidad del lenguaje, como suele ocurrir, es altamente probable que alguien lo haya implementado y subido de manera abierta a la comunidad. Efectivamente, basta con escribir en *google* '*python tree structure*' para encontrar entre los primeros resultados a *treelib*, un repositorio de *GitHub* con una eficiente implementación de esta estructura de datos [git11].

treelib

treelib es un proyecto sin finalizar. De todas formas, cuenta con las funcionalidades necesarias para generar una representación de la jerarquía de sectores que cumpla varios de los requisitos definidos anteriormente:

- Cuenta con dos clases: *Tree* y *Node*.
- Cuenta con los métodos típicos para crear, eliminar y modificar los nodos. Además, cada nodo cuenta con un atributo *data* donde poder almacenar los metadatos de cada industria.
- Cuenta con un método para realizar rápidas búsquedas entre los nodos del árbol y otros varios para recorrerlo.
- Cuénta con un método para visualizar el árbol generado e incluso escribir dicha visualización en el disco.
- Lamentablemente, no cuenta con la funcionalidad de guardar el árbol generado en disco de tal manera que luego pueda ser leído cómodamente por la misma librería. Las implicaciones que esto tiene es que se tendrá que volver a generar el árbol a partir del `.html` cada vez que

se arranque el *script*. No se contempla implementar el almacenado de la estructura dado que eso retrasaría el proyecto, pero es una propuesta para el futuro (sección 9.2).

Beautiful Soup

Para poder extraer la estructura jerárquica del fichero `.html` se utilizará la librería *Beautiful Soup*. Está escrita en *Python* y encaja perfectamente con el problema que se quiere resolver ya que cuenta con las siguientes características [Cru12]:

El funcionamiento es muy simple, primero se carga todo el fichero `.html`. Una vez cargado en la librería, son muchas las alternativas para realizar el análisis. En concreto, la funcionalidad a la que más provecho se le sacará es la de buscar (de fuera a dentro del código) el primer elemento `HTML` que cumpla unos requisitos o mejor dicho filtros que previamente se hayan definido.

Por tanto, primero se analizará visualmente el código de la página web para ver de qué manera se ha implementado en ella la estructura jerárquica de las industrias. En la figura 5.4 se muestra una captura del proceso de análisis visual utilizando para ello la herramienta *Inspector de elementos* del navegador web *Chrome*. Sea cual sea el diseño del código para representar la jerarquía, para construir el árbol en *Python* se necesitará primero encontrar la el nodo raíz (cuyo nombre es '*All Markets*') y partir de ahí encontrar la manera de ir descendiendo y crear el resto de nodos.

A primer vistazo parece que la jerarquía esté estructurada de tal manera que los elementos correspondientes a las industrias más grandes engloban totalmente a los más específicos, como se muestra en la siguiente simplificación:

Código 5.2: Ejemplo de estructura esperada del árbol

```
<div Clean Energy>
    <div Biofuels></div>
    <div Wind></div>
    <div Solar>
        <div Residental Solar></div>
        <div Commercial Solar></div>
    </div>
</div>
```

No obstante, al mirar el código más en detalle resulta que la estructura no es exactamente así. En realidad, cada sector está compuesto de dos elementos `HTML` hermanos entre sí, conteniendo el primer tipo la información del mismo (nombre y metadatos) y correspondiéndose el segundo con todos los elementos de otros sectores que tenga anidados (si no tiene ninguno el elemento estará vacío, pero existirá como tal). De hecho, ésta variación hace que la implementación de la función recursiva pierda elegancia (como se puede observar en el algoritmo 1 más adelante).

Estos dos elementos son los que se irán recorriendo por el árbol, y las acciones a realizar cuando el *script* se encuentre con uno o con otro serán distintas, por lo que es fundamental encontrar algo que les diferencie para guiar adecuadamente la construcción del árbol. Como es normal en elementos de `HTML` (*divs*[4], en este caso), cada uno tiene atributos distintos. En este caso basta con indicarle a *BeautifulSoup* que reconozca cada uno fijándose en el atributo *class* e *id* para poder reconocerlos perfectamente:

[4]Elemento `HTML` cuyo cometido es ser un contenedor unitario que encapsula otros elementos `HTML` y divide la estructura de la página web en secciones.

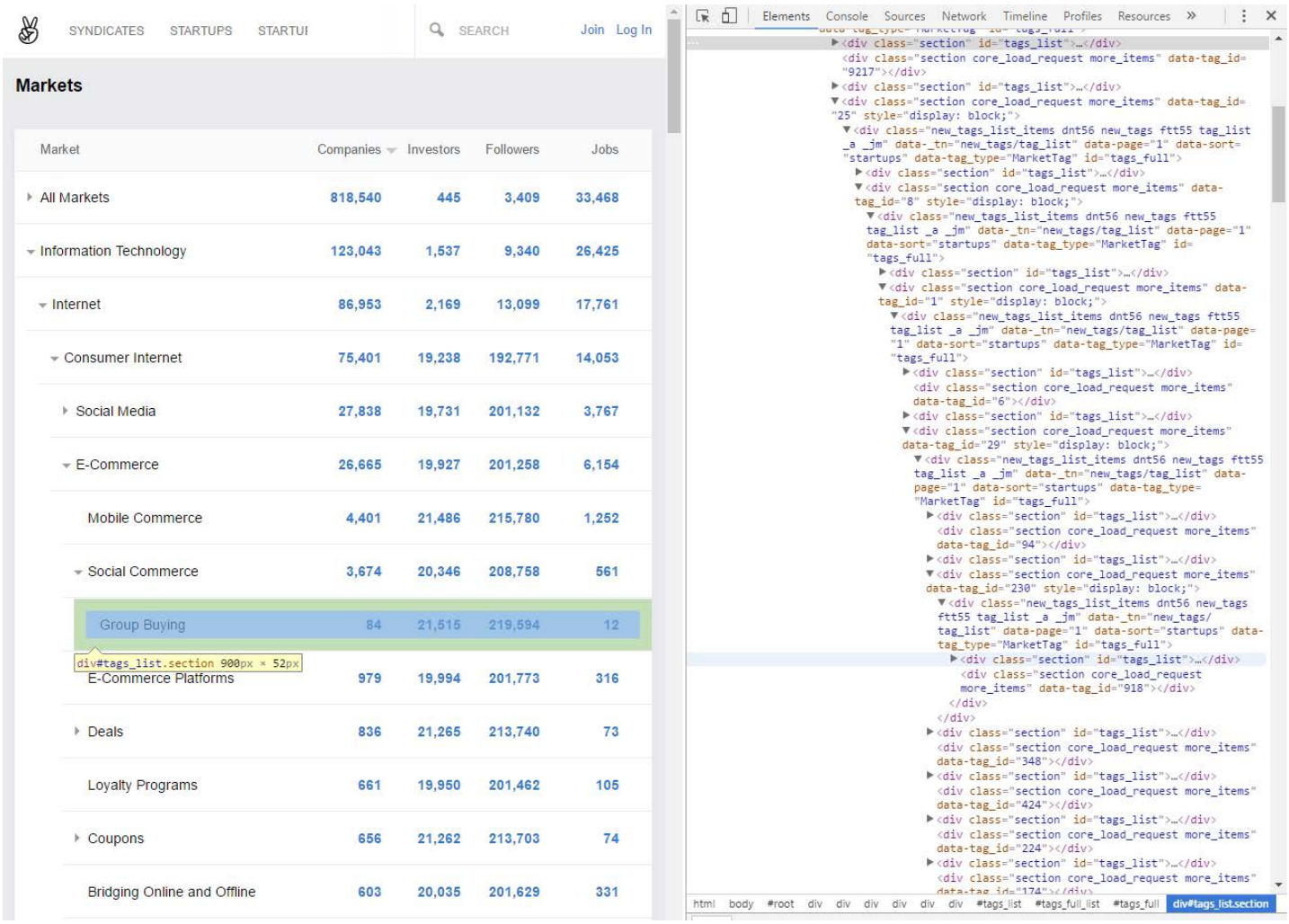


Figura 5.4: Explorando la estructura `HTML` de `https://angel.co/markets`

Código 5.3: Tipos de elementos HTML relevantes

```
<!  tipo 1 >
<div class='section' id='tags_list'></div>
<!  tipo 2 >
<div class='section core_load_request more_items'
    style='display: none;'></div>
```

Respecto al tipo 2 se recoge también el atributo *style* que ayudará a reconocer si el elemento está vacío o contiene información de sectores anidados. Podría representarse esta variación como un tercer tipo, pero se omitirá para hacer más clara la explicación.

Teniendo en cuenta todo lo dicho, en el código 5.4 se muestra un ejemplo de la estructura real del código `HTML`. Obviamente, esta muestra es una simplificación del código real. De hecho, en el código real existen además multitud de otros elementos `HTML` con características diferentes junto con fragmentos de *Javascript* y *JQuery* que no tienen información útil para lo que se quiere resolver. Por supuesto, esto no supondrá un problema para *BeautifulSoup* ya que éste sólo devolverá lo que se le pide ignorando el resto.

Cada vez que se desea insertar un nodo en el árbol, es necesario indicar el contenido del mismo (id **único**, nombre de la inustria y metadatos) y una referencia a su nodo padre (por id). Para mantener en todo momento del proceso de construcción **recursivo** del árbol dicha referencia al nodo padre se ha utilizado una pila[5]: antes de descender para explorar un nodo (ver si tiene hijos) se introduce mediante *push()* su id de tal manera que si efectivamente tiene hijos, éstos puedan saber quién es su padre y por lo tanto puedan ser insertados en el árbol simplemente consultando el último elemento insertado en la pila. El uso de la pila resulta especialmente útil cuando se ha alcanzado un nodo hoja y es necesario retroceder (ascender) para seguir explorando sin perder la referencia de quién es el padre del siguiente nodo.

En el algoritmo 1 se describe mediante pseudocódigo de qué manera se ha puesto todo lo anterior en común para finalmente implementar un algoritmo capaz de construir el árbol.

[5]Estructura de datos que sirve como colección de elementos *LIFO* (*last in, first out*) con dos operaciones: *push*, que añade un elemento, y *pop*, que saca el último introducido.

Código 5.4: Estructura real del árbol

```
<div tipo = 1>
        <! Clean Energy + metadatos >
</div>
<div tipo = 2>
        <div tipo = 1>
                <! BioFuels + metadatos >
        </div>
        <div tipo = 2>
                <! VACIO >
        </div>
        <div tipo = 1>
                <! Wind + metadatos >
        </div>
        <div tipo = 2>
                <! VACIO >
        </div>
        <div tipo = 1>
                <! Solar + metadatos >
        </div>
        <div tipo = 2>
                <div tipo = 1>
                        <! Residental Solar + metadatos >
                </div>
                <div tipo = 2>
                        <! VACIO >
                </div>
                <div tipo = 1>
                        <! Commercial Solar + metadatos >
                </div>
                <div tipo = 2>
                        <! VACIO >
                </div>
        </div>
</div>
```

Algorithm 1 Construcción del árbol sectorial

```
1:  procedure buildTree(path)
2:      html ← read(path)                                  ▷ cargar fichero html
3:      soup ← BeautifulSoup(html)                         ▷ parsear fichero html
4:      markets ← soup.find(2)                             ▷ el elemento donde empieza todo es de tipo 2
5:      tree ← tree(Node( 'root'))                         ▷ Inicializar árbol
6:      stack ← list()
7:      stack.push('root')
8:      createNodes(markets)
9:      return tree
10: procedure createNodes(content)
11:     if content.isType(1) then
12:         info ← content.getInfo()                       ▷ extraer industria y metadatos
13:         tree.createNode(info,stack.last())             ▷ createNode(data,parent)
14:         stack.push(info.id)                            ▷ descender un nivel
15:     else
16:         if content.isType(2) then
17:             if content.notEmpty() then
18:                 firstBorn ← content.find(1)
19:                 createNodes(firstBorn)
20:                 for all sibling ∈ firstBorn.getSiblings() do
21:                     createNodes(sibling)
22:             stack.pop()                                ▷ ascender un nivel
```

A continucación se listan y describen dos aspectos importantes relativos a la construcción del árbol que no se consideraron en el pseudocódigo 1 por no ser propios de la algoritmia del *script* pero si de la implementación en sí:

- **Controlando el flujo de ejecución con excepciones:** En la función *createNodes()* de la implementación real se han rodeado sendos bloques del *if-else* principal con sentencias *try-except* ya que las condiciones se evalúan para varios elementos `HTML`, lanzando algunas excepciones dado que no todos cuentan con los atributos cuyo valor requiere ser comprobado (elementos que no tienen información útil respecto al árbol).

  Se podría argumentar que las excepciones no son errores, y que deberían ser usadas únicamente en situaciones excepcionales (por ejemplo al intentar escribir un fichero a disco y no queda más espacio o no se tienen permisos) y no para controlar el flujo de ejecución como se hace aquí. Ésta concepción de las buenas prácticas en el uso de excepciones es muy común en muchos otros lenguajes como por ejemplo *C* con su estilo de codificación '*Look before you leap*' [Fou91b] pero no en *Python*, donde el principio que rige es '*It is better to ask for foregiveness than permission*' [Fou91a], recomendándose en este último caso el uso de dichas sentencias para controlar el flujo de la ejecución.

- **Extrayendo el texto visible:** Cuando la función *createNodes()* da con un elemento `HTML` de tipo 1 significa que tiene que extraerle el nombre de la industria y los metadatos. Como muestra la imagen 5.5, el elemento en sí tiene esa y más información anidada y estructurada

utilizando otros elementos `HTML`. Nótese además que el código de la figura 5.5 ni siquiera está desplegado del todo:

Figura 5.5: Elemento HTML tipo 1 real

Existen pues dos opciones para separar la información que se quiere (nombre y metadatos) de la que no (código `HTML` y enlaces):

(a) Analizar visualmente el código para detectar qué características tienen los elementos que contienen la información requerida (nombre de la industria y metadatos) para a continuación utilizar los métodos *find()* (como se hizo en las líneas 4 y 18 del algoritmo 1) y similares que ofrece la librería *BeautifulSoup* para encontrarlos de manera precisa y poder acceder directa y exclusivamente a la información clave.

(b) Utilizar la función *find_all(text=True)* de *BeautifulSoup* que devuelve una lista con todos los textos **visibles** que contiene cada elemento perteneciente a un fragmento de un código `HTML`.

La primera opción resulta viable, no obstante, cada porción de información que se quiere extraer está en varios elementos `HTML` diferentes y hacer una regla para cada uno de ellos sería preciso pero algo tedioso y engorroso. Por otro lado, prácticamente todo el texto visible en los elementos de tipo 1 es justamente la información que se quiere extraer, por lo que se optará por la segunda opción. Los textos visibles no útiles que también son recogidos aquí son simplemente saltos de línea y tabuladores generados entre los elementos `HTML`, que se limpian muy fácilmente.

Una vez construido el árbol se puede exportar una visualización del mismo a un archivo `txt` utilizando la función *save2file()* implementada en *treelib*. Como contiene muchas industrias, en la figura 5.6 se muestra un pequeño fragmento del mismo.

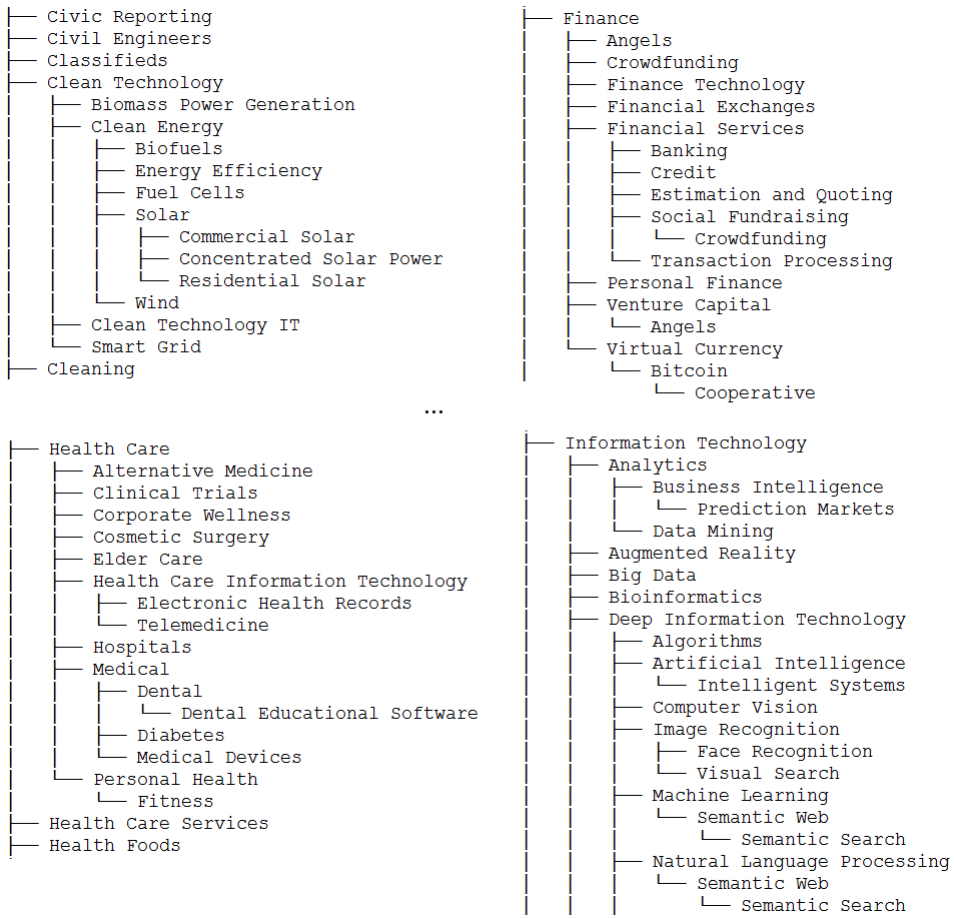


Figura 5.6: Cuatro fragmentos distintos del árbol de industrias

Hay varios elementos del árbol que dudablemente puedan ser considerados como una industria en sí. En cualquier caso, dichos elementos, llámense sectores, industrias, especializaciones o meramente palabras clave ofrecen una información ciertamente precisa respecto a la actividad de la empresa y eso es justo lo que se pretendía conseguir (como se expuso en la fase del entendimiento del negocio de la subsección 5.1.1).

Reglas

Ahora que se dispone del árbol de industrias perfectamente cargado en memoria y preparado para ser utilizado, se implementará una de las funcionalidades más importantes: la búsqueda en el texto de entrada de las palabras clave existentes en el árbol y la asignación de los nodos padre para completar toda su relación semántica.

- **Búsqueda con sinónimos**: la búsqueda exclusiva en el texto de fragmentos como '*clean technology*' o '*e-learning*' para recoger estas actividades resultaría en muchos falsos negativos,

ya que las descripciones con '*cleantech*' o '*elearning*' no serían etiquetadas. Para minimizar los falsos negativos (convirtiéndolos en verdaderos positivos) hay que realizar un trabajo algo más elaborado mediante el uso de sinónimos. Por tanto, se generará un fichero `.csv` donde habrá una línea con todos los nombres de los nodos del árbol. Manualmente se añadirán sinónimos en cada fila según el caso. En el cuadro 5.3 se muestran algunos ejemplos. Como se ve, la estrategia de búsqueda de cada término varia:

- Primero se busca la industria/palabra clave tal cuál. Si efectivamente se encuentra el nombre de la actividad es suficiente para etiquetar y no es necesario buscar por los términos de búsqueda ('*design*').
- En ocasiones hay que rodear el término de búsqueda con espacios para evitar falsos positivos. Por ejemplo si se buscase '*art*' o '*iot*' en lugar de ' *art* ' o ' *iot* ' se podrían etiquetar erróneamente aquellas descripciones que contuviesen palabras como '*p***art***icularly*' o '*b***iot***echnology*', respectivamente.
- Hay veces que la relación semántica establecida entre la industria/actividad y los términos de búsqueda no es de estricta sinonimia pero aun así se establece porque se asume que al hacerlo el incremento de verdaderos positivos es mayor que el de falsos positivos ('*data mining*' ←'*machine learning*' | '*deep learning*').
- Hay veces que varios posibles términos de búsqueda se simplifican en uno sólo que no tiene significado propio pero conforma una cadena que identifica la actividad ('*crowdourc*') de la empresa.

Industria/palabra clave	Término de búsqueda
'clean energy'	'cleantech','clean tech'
'data mining'	'machine learning','deep learning'
' art '	'museum','artist'
'internet of things'	' iot '
'm2m'	'machine to machine'
'design'	
'crowd-sourced labor'	'crowdsorc'

Cuadro 5.3: Reglas de búsqueda de industria

En total se tienen alrededor de 1800 industrias/palabras clave. Primero se computó el listado de las industrias existentes en el árbol escribiéndose éste en un fichero `.csv` de una única columna para posteriormente ir rellenando con sinónimos manualmente según se requiera en la segunda columna inicialmente vacía.

- **Asignación jerárquica** Una vez que se ha identificado la existencia de un nodo del árbol de actividad sectorial en el texto, se etiquetará a la empresa con el identificador del nodo. Pero además, también será etiquetada con todos los identificadores de los padres de dicho nodo. Así, si se hace *match* con *Bitcoin*, se añadirá *Payments* y también *Finance*, y si se hace con *Android development*, se añadirá *Software* e *Information Technology*. Dado que se tiene la estructura arbórea completamente cargada en memoria, la manera óptima de realizar estas asignaciones en mediante llamadas recursivas a al método *getParent(node)* que devuelve el identificador del padre inmediato de *node*.

5.2.4. Evaluación

En primer lugar, cabe destacar que se han cumplido todos los requisitos de la tabla 5.2. En concreto, el 06 y el 09 se han cumplido fácilmente comprobando en tiempo de ejecución mediante la llamada a *platform.system()* en qué SO se estaba ejecutando (dado que había algunas líneas de código que no podían ser iguales). Por otro lado, el cumplimiento del 05 se explicará en la sección 8.1.

Al ser los sectores/industrias nuevos frente a los del dataset del que se dispone, no es posible automatizar esta fase, por lo que la evaluación tiene que ser también manual:

1. Ejecutar el *script* y analizar visualmente 50 muestras aleatorias de descripciones con industrias asignadas y ver si el etiquetado tiene sentido o no.

2. Anotar aquellas industrias que se repiten mucho sin sentido (potenciales falsos positivos).

3. Corregir el fichero de reglas.

4. Si ha habido correciones, ir al paso 1, sino, finalizar.

La mayoría de correcciones efectuadas fueron originadas por buscar términos demasiado genéricos en el texto, como '*art*' o '*it*' que suelen aparecer dentro de muchas palabras. La manera de corregirlas no es igual siempre. En el caso de '*art*' basta con añadir espacios delante y atrás, pero no es el caso de '*it*' (abreviación de *Information Technology*), que coincide con el pronombre 'ello' y por lo tanto se eliminó. La figura 5.7 contiene las mismas descripciones que la 5.3 pero utilizando este extractor de reglas + árbol. Como se puede observar, el etiquetado realizado consigue una imagen mucho más completa de la actividad de la empresa. Hay algunos falsos positivos, pero no son muchos por lo que se concluye que la solución obtenida cumple con los objetivos de este capítulo.

```
_____________________________________________________________________
1, id: 293044
      Description:
Fenway Summer is a consumer finance advisory and investment firm
headquartered in the historic Georgetown neighborhood of Washington,
DC.
      industries: ['finance', 'advertising', 'investing',
'enterprises', 'consumers']
_____________________________________________________________________
2, id: 589533
      Description:
Seventh Continent, a business game platform, enables the exchange of
digital commodities created and traded in bitcoin by virtual
companies.
      industries: ['e-commerce', 'bitcoin', 'gaming', 'platform',
'bitcoin', 'virtual currency', 'payments', 'trading', 'information
technology', 'finance']
_____________________________________________________________________
3, id: 349539
      Description:
Design your own fashion design fashion
      industries: ['e-commerce', 'design', 'fashion', 'clothing']
_____________________________________________________________________
4, id: 672569
      Description:
Generate more mobile leads and sales with Viva Tags. Our mobile
commerce platform simplifies mobile marketing, purchases, and
payments. Alcohol & payments.
      industries: ['marketing', 'finance', 'alcohol', 'food &
beverages', 'deals', 'mobile', 'payments']
_____________________________________________________________________
```

Figura 5.7: Resultados del extractor con reglas + árbol

Dada la potencia de esta trabajada herramienta, se utilizará en otros proyectos fundamentalmente para lo que sirve, identificar la actividad de la empresa a partir de su texto. Además, una ventaja de esta herramienta es que se puede enriquecer/corregir de manera fácil y continua. Este tipo de mejoras para el futuro se harán en la sección 9.2.

5.3. Matching con sectores

En algunas partes del trabajo será necesario incorporar conocimiento experto externo a los modelos que se vayan creando. Existen algunos indicadores financieros a nivel de empresa que, como se explicará posteriormente, son de alto interés para el método de valoración. El acceso a los mismos es demasiado difícil (especialmente con *startups*). En esas situaciones, se utilizarán las medias sectoriales de dichos indicadores. En futuros apartados se comentará hasta que punto es correcto o no suponer que estos indicadores no presentan grandes desviaciones típicas dentro de las empresas de un mismo sector. El caso que aquí concierne es que cuando se quiera incorporar los promedios de los indicadores al dataset existirá un problema de *matching* ya que los sectores/industrias de una fuente se llamarán potencialmente diferente en las otras.

Aswath Damodaran es un prestigioso y activo profesor de finanzas especializado en valoración

de empresas. Damodaran actualiza anualmente una sección de su página web[6] donde recopila varios indicadores financieros (medias de industrias para EEUU y compañías internacionales) así como distintas métricas de valoración que el mismo calcula [16a]. Será útil incorporar esas medias al dataset. Damodaran hace las medias para 95 industrias. En este proyecto se trabaja con aproximadamente 1800 etiquetas de actividad económica, por lo que merece la pena dedicar un esfuerzo para que el *matching* sea lo más preciso posible.

La idea es implementar un *script* capaz de asignar automáticamente uno o varios sectores de Damodaran a un conjunto de *tags* de una empresa previamente etiquetada mediante el extractor implementado en la sección anterior. Lo primero será descargar los sectores de Damodaran y cargarlos en *Python*. A continuación se describen los siguientes pasos a realizar:

1. Pasar el extractor de palabras clave a la columna de sectores de Damodaran generando una nueva columna *tags*. La idea es que la existencia de estos *tags* en la descripción de una empresa implica que existe cierta relación con el sector de Damodaran.

2. Escribir en un fichero `.csv` ambas columnas (sectores, *tags*), abrirlo con un editor de texto y empezar a analizarlo línea a línea visualmente.

3. Eliminar del campo *Tags* aquellas etiquetas que sean demasiado genéricas como para implicar que si esta está presente entonces el sector correspondiente es el indicado en el campo de Damodaran.

4. Hacer las modificaciones pertinentes en los ficheros de *matching* locales (palabras clave, industrias) si se ve alguna oportunidad de mejora.

5. Ejecutar la función de *matching* sobre una columna de un dataset con *tags* previamente extraídos y evaluar/verificar que las asignaciones de los sectores de Damodaran tienen sentido, efectuando correcciones en el fichero de *matching* si se detecta alguna imprecisión

6. Analizar qué conjuntos de *tags* se han quedado sin sector asignado y por qué. Si es necesario, modificar el fichero de *matching* para que se les asigne algun sector.

7. Si ninguno de los *tags* de una empresa hace *match* con algún sector de Damodaran el *script* le asignará un valor por defecto que se corresponderá con el indicador financiero pertinente medio del mercado (también calculado por Damodaran).

La función obtenida recibe un conjunto de etiquetas (generadas por el extractor) y devuelve un conjunto de sectores de Damodaran (normalmente uno), mediante los cuáles podemos indexar los distintos indicadores sectoriales y combinarlos de la manera que se estime oportuno.

[6] `www.pages.stern.nyu.edu/~adamodar/`

Capítulo 6

Tasa de descuento y riesgo

En este capítulo se explicará la manera en la que se han tenido en cuenta e introducido los factores relativos a la tasa de descuento (r_W y g, WACC y crecimiento respectivamente) así como la manera en la que están distribuidos los componentes del riesgo en los inputs de la fórmula (SR o *Success Rate* y WACC).

Primero se abordará la tasa de descuento, empezando por el crecimiento y a continuación el WACC. Del intento de estimar este último se desprenderá la necesidad de crear el input SR.

Para facilitar al lector el entendimiento del proceso se reescribe la fórmula 2.4 en los términos relevantes a este capítulo:

$$V = \frac{SR \times FCF}{r_W - g} \tag{6.1}$$

donde:

- V: Valoración monetaria de la empresa.
- r_W ó $WACC$: *Weighted Averaged Cost of Capital* ó coste medio ponderado del capital.
- SR: *Success Rate* ó probabilidad de éxito (no quebrar).
- FCF: *Free Cash Flows* ó flujos libres de caja.
- g: *Growth* ó crecimiento anual de los flujos de caja libres esperados.

6.1. Crecimiento

6.1.1. Entendiendo del negocio y objetivos

Una parte importante de de la valoración de una empresa viene determinada por el crecimiento de sus activos. La falta de históricos de cada una de las empresas del dataset hacen de este un importante problema.

Una posibilidad consiste en almacenar un *timestamp*[1] cada vez que se actualice algún dato de una empresa en el dataset. La idea sería tener suficientes valores para poder construir un histórico a

[1]Secuencia de información relativa a un determinado evento a lo largo del tiempo

partir del cuál inferir una tasa de crecimiento relacionada directamente con la empresa. Se podrían tomar *timestamps* de los siguientes campos:

- Número de empleados.
- Número de seguidores en redes sociales.
- Número de descargas de aplicaciones (en el caso de que la empresa tenga una).
- Número de visitas a su página web.
- Los ingresos regresados como se explicó en la sección 7.1.

Habría que esperar varios meses para obtener una ventana temporal con suficientes puntos. Dado que se pretende que el método sea capaz de valorar empresas muy jóvenes, se plantea que a partir de 7 meses se podría empezar a inferir históricos.

De momento, habrá que idear otra estrategia para estimar el crecimiento con los datos que están actualmente al alcance. Como se comentaba al principio de esta sección, la tasa g de la fórmula 6.1 debe representar un crecimiento de los activos del negocio, más concretamente se debería aproximar a cómo van a ir evolucionando los flujos de caja a lo largo del tiempo. Resulta inviable (de momento) realizar una medida *adhoc* de las compañías para inferir el crecimiento o decrecimiento de sus ventas, pero lo que sí que se puede plantear es averiguar cuál es el crecimiento del interés del mercado en su producto. Dado que se ha conseguido hacer un retrato de la actividad de la empresa (capítulo 5) a base de etiquetas, se propone extraer de alguna manera el crecimiento de la popularidad de estos *tags* por parte del público general.

Los históricos de interés que se obtendrían serían del *tipo* de producto y no del producto en sí (de momento se tiene información precisa de los productos del 32 % de las compañías). Es conveniente reconocer que este hecho implica que el indicador del histórico sería una medida realmente optimista del interés del público y por tanto de las ventas de la empresa. Para amortiguar este efecto se deberá diseñar una fórmula que cuantifique moderadamente el gráfico obtenido.

Los objetivos quedan definidos entonces como:

1. Descargar los datos de las tendencias.
2. Diseñar una fórmula que interprete los datos de tal manera que su resultado sea una estimación del interés por el público del tipo de producto que ofrece la empresa (que se asumirá como g).

6.1.2. Obtención de los datos

Se utilizará una fuente de datos de la propia empresa que cumple precisamente la funcionalidad requerida. Consiste en que el usuario introduce un término y el sistema devuelve una curva que muestra cómo ha ido creciendo el interés de la gente respecto a esa palabra clave a lo largo de los últimos años. El intervalo se puede definir, así que se cogerán datos desde 2004 hasta el día presente (es un intervalo suficientemente amplio para analizar el interés del público a lo largo del tiempo).

Por lo tanto, será necesario descargar las curvas correspondientes a todo el conjunto de palabras clave, sectores e industrias con cierta periodicidad. Se estima que, dado que el crecimiento a estimar es anual, bastará con que el *script* a implementar descargue los datos una vez cada mes para tener los datos actualizados.

6.1.3. Entendiendo y preparando los datos

Primero, es conveniente aclarar una limitación importante de los datos descargados: los niveles de popularidad de un término no son absolutos, sino relativos a la propia popularidad del término en los distintos instantes temporales. Los resultados descargados no tienen ningún sentido cuantitativo. Para cada posible término, siempre hay algún instante con valor 0 (el mínimo), que indica que el momento en el que menos interés existía por parte del público a lo largo del periodo, y otro con 100 (el máximo), significando el momento de máximo interés. Esta limitación supone que no se puede medir si un término es más o menos popular que otro. Afortunadamente, el principal objetivo de esta sección es medir precisamente el crecimiento de la popularidad y no la popularidad en sí misma, por lo que esta limitación no supone ningún problema para el proyecto.

Dado que se han obtenido los datos directamente, no es necesario llevar a cabo un preproceso importante de la información, así que no será necesario utilizar *BeautifulSoup* (no hay código `html` que deba ser analizado). A continuación se muestra una pequeña parte de uno de los ficheros descargados:

```
From:   2004
To:     2016
Keyword: artificial intelligence

Day Interest
2004-01-04 64
2004-01-11 71
2004-01-18 82
2004-01-25 85
2004-02-01 80
2004-02-08 75
2004-02-15 73
2004-02-22 69
...
```

Como se puede apreciar, está en un formato bastante digerible para un programa informático. También se observa que cada punto de la gráfica se corresponde con una semana, y hay tantos puntos como semanas desde Enero de 2004 hasta hoy.

En resumidas cuentas, el trabajo de preparación requerido es escaso y sólo hay que limpiar algunos ejemplos que al no tener resultados generaron ficheros vacíos. Por la tanto, el *script* encargado de leer los ficheros y convertir los datos a un objeto *Pandas* tipo *Series* (donde el índice es la fecha y el valor el interés) es muy sencillo.

6.1.4. Procesando los datos

Una vez que se tiene implementado el programa que carga todo en memoria, el *script* ha de ser capaz de convertir las curvas correspondientes a un sólo indicador que efectivamente represente el crecimiento del interés, g, para un término de búsqueda k. La fórmula que a priori parece razonable usar es la del crecimiento medio:

$$g(k) = \frac{P_t(k) - P_{t-1}(k)}{P_{t-1}(k)} \tag{6.2}$$

donde P_t es la popularidad relativa en el instante t.

No obstante, esta fórmula tiene ciertas limitaciones en el contexto del problema presente:

- Las variaciones de popularidad del pasado lejano tienen el mismo peso que las del cercano. Lo adecuado sería que los incrementos/decrementos de popularidad que ocurrieron hace 8 años tuviesen menos peso que los que tuvieron lugar hace 1 mes.

- Sólo se miden dos puntos. Por tanto, las variaciones de popularidad que se suceden entre *t-1* y t no computan de ninguna manera. Sería beneficioso que el indicador g también estuviese influenciado por las variaciones que ocurran dentro del intervalo a medir.

Para solventar estos aspectos se ha desarrollado una nueva fórmula. La idea subyacente es dividir la linea temporal en *iterations* trozos y calcular la variación media por semana en cada uno de ellos. Después, se calculará una media ponderada de las *iterations* variaciones anteriores, dando más peso a los intervalos más cercanos al momento presente. Formalmente:

$$g = \sum_{i=0}^{iterations} \left(\frac{P_{c(i+1)} - P_{ci}}{c} \times \frac{i}{iterations} \right) \tag{6.3}$$

donde:

- $c = \left\lceil \frac{n\_weeks}{iterations} \right\rceil$, es decir, el tamaño de los intervalos medido en número de semanas.

- n_weeks es el número de semanas total del intervalo descargado.

Se podría pensar que lo razonable hubiese sido ponderar el término de la ecuación 6.2 por cada intervalo en lugar de las variaciones medias por semana. No obstante, se ha hecho así porque:

- La popularidad P no mide unidades absolutas de popularidad, sino relativas, por lo que sí tiene sentido dividir por c.

- Los resultados obtenidos son más satisfactorios.

6.1.5. Evaluación y ejecución

El término *iterations* es una constante de la fórmula. Cuanto más grande sea más pequeño será c y más se tendrán en cuenta las variaciones de interés intermedias. Pero si se establece un valor demasiado alto, puede hacer que el indicador pierda su sentido. Tras efectuar algunas pruebas se determina que 70 es el valor adecuado para *iterations*. Por otro lado, los valores que devuelve g son muy cercanos a 0 por lo que se decide multiplicar el resultado por 200.

La figura 6.1 muestra la ejecución del *script* con los parámetros ajustados a los valores anteriores para algunos ejemplos de términos de búsqueda.

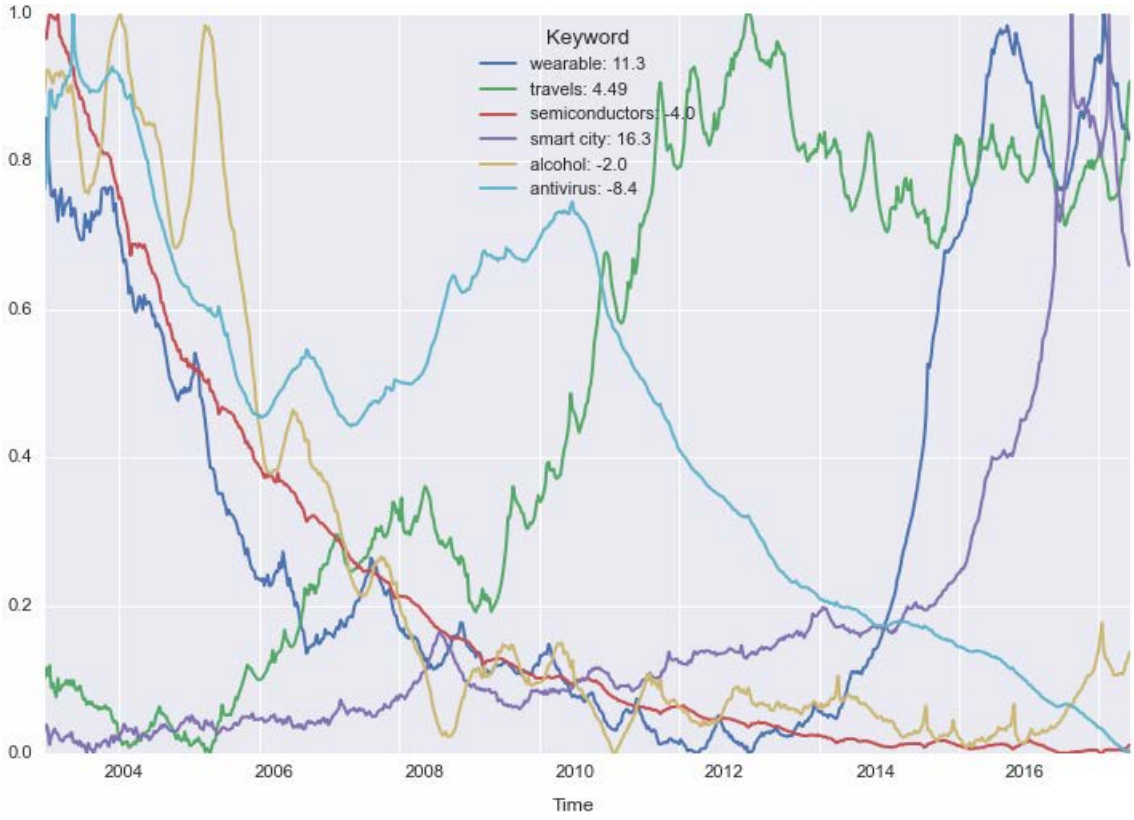


Figura 6.1: Tendencias de *keywords* con su puntuación asociada

Como se puede observar en la imagen 6.1, la función diseñada cumple con los objetivos explicados en el apartado 6.1 y solventa satisfactoriamente las características no deseadas que hubiese implicado utilizar la fórmula 6.2.

Ahora bien, todavía hay que abordar un problema. La función 6.3 da un indicador del crecimiento en base a la tendencia de una *keyword*, pero la actividad de una empresa está definida por más de una. Tendiendo esto en cuenta, la solución implementada obtendrá todas las tendencias de las etiquetas de una empresa primero y calculará la media después.

6.2. WACC

En la sección 2.2 ya se introdujo que la tasa a la que se van a descontar los flujos es $r_W - g$, por lo que, dada la importante repercusión de la misma en la valoración, habrá que definir claramente como se van a estimar sus componentes. En concreto, es el objetivo de esta sección estimar el WACC o r_W.

Se puede obtener una visión algo más completa de qué es realmente el WACC consultando la bibliografía. Resulta especialmente ilustrativa la aclaración que hace el profesor de Finanzas Pablo

Fernández acerca de que el WACC es un promedio ponderado de dos magnitudes muy diferentes [Fer11]:

- Un coste: el coste de emitir deuda (r_D).
- Una rentabilidad exigida a las acciones (r_E). Aunque a r_E se le denomina con frecuencia 'coste de las acciones', existe una gran diferencia entre un coste y una rentabilidad exigida.

La conclusión es que el WACC, al contrario de lo que sugiere su nombre, no es ni un coste, ni una rentabilidad exigida.

Ahora bien, el objetivo principal del proyecto es poder valorar a empresas no demasiado grandes y *startups*. Es especialmente atípico en este último tipo de organización encontrarse con emisión de deuda (siguiendo con lo expuesto en la sección 2.3). Por tanto, en este problema lo habitual será valorar empresas cuyo único instrumento de financiación es el *equity* y en consecuencia no será un error afirmar que el WACC representará lo que los inversores *exigen* a las acciones (r_E).

Como se expuso en la sección 2.3, existen ciertas limitaciones en este proyecto por lo que no se aplicarán las técnicas tradicionales de estimación del WACC: ya sea calculando cada uno de sus componentes por separado (como se mostró en la sección 2.2) u otros cálculos habituales que implican el conocimiento de betas[2] o las desviaciones típicas de los retornos del *equity*, factores que difícilmente pueden entrar en el cálculo de valoraciones de *startups* [Dam09].

Por otro lado, se puede dar una lectura a este problema como una estimación de las funciones de utilidad[3] de los agentes del mercado (inversores principalmente) respecto a ciertos activos financieros (empresas/*startups*). En este proyecto, se asumirá la existencia de un agente representativo. Es decir, se asumirá que los agentes del modelo económico subyacente tienen funciones de utilidad distinta pero actúan de tal manera que la suma de sus preferencias es matemáticamente equivalente a la decisión de un único individuo o conjunto de individuos idénticos. Esta suposición de que el mercado cuenta con mecanismos internos para coordinar los dispares intereses de millones de agentes que de alguna manera recuerda al fenómeno de la 'mano invisible' [4], aunque ampliamente utilizada, ha sido criticada desde el punto de vista macroeconómico [Kir92]. No obstante se tomará como válida ya que proporcionará una base útil para la estimación de los componentes del WACC y encaja muy bien con las limitaciones del problema que se quiere resolver (sección 2.3). Siguiendo en la línea de la suposición de la existencia de un agente representativo, se asume que el mercado busca optimizar *exclusivamente* dos factores a la hora de ponerle precio a las empresas: la media y la varianza de los flujos de caja que genera una compañía. Es decir, que si un inversor tuviese ante sí todas las opciones de inversión del mundo él intentará siempre encontrar la mejor relación media-varianza de los flujos generados por cada activo en cuestión. Por tanto, a la hora de calcular la valoración de una compañía, habrá que extraer los momentos correspondientes a la media y la varianza para la función de utilidad del agente representativo que le 'llegan' al inversor dadas las características de dicha empresa.

En la sección 2.2 se introdujo el papel del riesgo como componente en la tasa de descuento del modelo de Descuentos de Flujos de Caja para estimar el valor intrínseco de una compañía.

[2] Mide la volatilidad o el riesgo sistemático de la empresa en comparación con el mercado.

[3] Expresión del deseo en términos matemáticos. En el contexto del problema se diría que expresa las preferencias de los inversores en función de la rentabilidad y el riesgo de los distintos productos de inversión.

[4] Término utilizado por *Adam Smith* para expresar su noción de que los esfuerzos de los individuos en perseguir sus propios intereses a menudo benefician más a la sociedad que si sus acciones fuesen dirigidas precisamente a beneficiarla [Smi76].

Siguiendo esta línea resulta conveniente citar al profesor Aswath Damodaran en su *Darkside of Valuation*[Dam01]:

> 'Discounted cash flow valuation is a tool for estimating intrinsic value, where the expected value of an asset is written as the present value of the expected cash flows on the asset, with either the cash flows or the discount rate adjusted to reflect the risk.'

En este método de valoración se tendrá en cuenta el riesgo idiosincrático[5] por un lado, entendido como SR o *Success Rate* mediante el ajuste de los flujos de caja (sección 6.3) y por otro se utilizará como tasa de descuento el WACC medio del sector menos la tasa de crecimiento ($r_W - g$). El WACC debe representar también el riesgo. Es importante recalcar que la naturaleza de dicho riesgo debe ser sistemática[6] y no idiosincrática, ya que el riesgo idiosincrático apenas tiene correlación con el mercado y por lo tanto es correcto suponer que los inversores lo eliminarán diversificando sus carteras [Goesd]. Por tanto, una opción válida será utilizar las medias del WACC calculadas por Damodaran (sección 5.3) de cada industria/sector. Esta será la estrategia a seguir inicialmente ya que es una manera correcta de tener en cuenta el riesgo sistemático de mercado al que pertenece cada compañía. Siguiendo el CAPM [Dama] (*Capital Asset Pricing Model*), el retorno que exigen los inversores a las acciones, osea, r_E, (si se asume que no hay deuda se podría hablar directamente del WACC) de una compañía será mayor cuanta más covarianza exista entre ella misma y el mercado:

$$r_E = r_f + \beta_E \times (E(r_m) - r_f)$$

donde

- r_E retorno de las acciones de la empresa.
- β_E es una medida del riesgo (volatilidad) no diversificable (sistemático) de las acciones. Se calcula como una covarianza estandarizada entre r_E y r_m.
- $E(r_m)$ es la esperanza matemática del retorno de la economía en su conjunto.
- r_f es la tasa libre de riesgo, que indica el retorno teórico de una inversión sin riesgo.

En general se cumple que las empresas de un mismo sector tienen una β_E similar y por lo tanto la rentabilidad que se les exige a sus acciones también es parecida ya que su precio responde de manera similar a los movimientos del mercado. Así, si las empresas biotecnológicas covariasen a la par que la economía en su conjunto ($\beta_E = 1$), entonces cuando el mercado subiese un 10 %, así lo harían sus acciones.

El modelo CAPM, aunque elegante, no es perfecto [Dama]. Un aspecto a tener en cuenta es que las β_E de la empresas más jóvenes de un sector suelen ser más altas que las de las compañías más maduras. Este hecho se investigó cuando el proyecto estaba en una fase avanzada, por lo que no se tuvo en cuenta durante la primera iteración. En la sección 9.2 de propuestas para el futuro se explicará una estrategia para atajar los problemas que pueden surgir a raíz de este desajuste.

[5] Riesgo específico para a un determinado (y pequeño) conjunto de activos.
[6] Riesgo inherente a un mercado o segmento de mercado.

6.3. Flujos de caja esperados

En esta sección se detallará el proceso mediante el cuál el modelo de valoración calculará el riesgo idiosincrático de una empresa. Dicho riesgo será utilizado posteriormente para calcular la esperanza matemática de los flujos de caja que genera una empresa. Esta media se corresponderá con la media del agente representativo y como se verá su cálculo no supone ningún problema, ya que se trata de una transformación lineal en la que es correcto asumir que la media de los agentes individuales es la media del agente representativo.

Existen varios artículos y estudios que hacen referencia a alto número de *startups* que fracasan[MHDB12]. Para analizar cuál es el riesgo de quiebra de las empresas se utilizarán los datos relativos a uno llevado a cabo por el *Statistic Brain Research Institute* [16b] en 2016 en el que se analizaba precisamente cuáles son los factores habituales que hacen que una empresa joven falle. En dicho informe se indican algunas causas principales interesantes relacionadas con la incompetencia de los directivos a la hora de ajustar el precio de sus productos y escasez de planificación o con la falta de aptitudes propias del *management*. Desafortunadamente, aunque este tipo de factores a menudo cualitativos sean ciertamente determinantes, es prácticamente imposible tener acceso a ellos de manera masiva. Por ejmplo, se podría pensar que podría ser interesante recopilar la carrera profesional de los fundadores de la empresa para ajustar mejor el riesgo en función de su historial y nuevo negocio. Se podría incluso puntuar la educación que las personas han recibido acorde a ranking de universidades, por ejemplo. Son múltiples las posibilidades, pero no pueden llevarse a cabo de momento. No sólo porque sea difícil acceder a dichos datos, sino porque recopilar información de carácter personal es no es legal[7] y se saldría del marco regulador de este proyecto. Por tanto, para estimar el riesgo se utilizará exclusivamente el efecto de la edad de la empresa y su actividad.

6.3.1. En función de la edad

De manera intuitiva se puede deducir que cuando más joven es una empresa, más riesgo de quiebra existe. Y efectivamente, es así.

Concretamente, en el estudio en el que se basa esta sección se provee del porcentaje de empresas que empiezan y llegan al primero, segundo, tercero y así hasta el décimo año.

[7] `http://ec.europa.eu/justice/data-protection/data-collection/legal/index_en.htm`

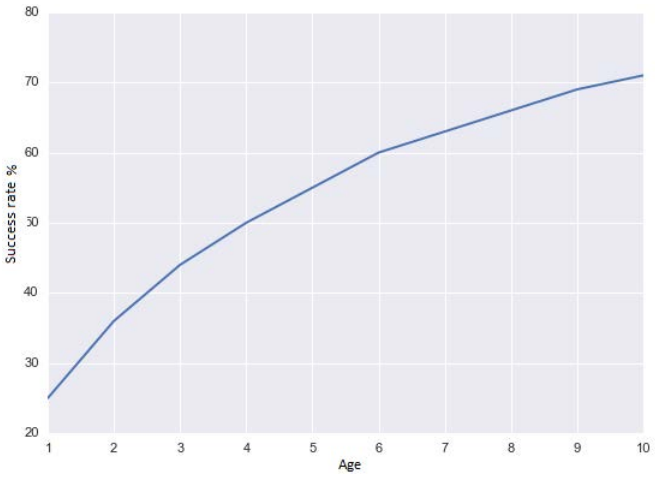


Figura 6.2: *Success rate* o probabilidad de no quebrar en función de la edad de la empresa

En la figura 6.2 se han dibujado los datos proporcionados y se aprecia claramente este efecto, manifestando además una tendencia logarítmica que significa que cada año que pasa, las probabilidades de sobrevivir otro año más aumentan ligeramente. Este fenómeno podría estar perfectamente relacionado con el hecho de que según va pasando el tiempo, cada vez hay menos competidores de la misma edad, siendo la mayoría startups muy jóvenes que suponen un nivel más bajo de amenaza.

6.3.2. En función de la actividad empresarial

El mencionado estudio provee además de cuál es el riesgo de quiebra de una empresa según el sector al que pertenezca. Lo cuál tiene mucho sentido, ya que cada uno de ellos está en un estado de maduración distinto y el comportamiento de los mercados internos será potencialmente diferente en los distintos casos.

No obstante, el estudio referenciando sólo expone los datos para el cuarto año. Es decir, el porcentaje de empresas de cada sector que alcanzan el cuarto año. Dado que la función final a obtener tiene que asignar una probabilidad de quiebra a cada par *(año,sector)*, se efectuará una transformación lineal combinando los datos de la sección 6.3.1 anterior con los de este.

Se define $P_{sector}(age)$ como la probabilidad de que el sector *sector* llegue a la edad *age*, siendo P_{avg} el promedio entre los distintos sectores (50,45 %). Como se ha dicho, en el estudio sólo se proporciona $P_{sector}(4)$. Por otro lado, se define $Failed(age)$ como el porcentaje de empresas que que no han conseguido llegar a la edad *age* (es una lectura que se puede dar a la función de la figura 6.2). Por tanto, la función definitiva se define como:

$$FailureRate(age, sector) = 100 - Failed(age) \times ratio_{sector} \tag{6.4}$$

donde:

$$ratio_{sector} = \frac{P_{sector}(4)}{P_{avg}(4)} \tag{6.5}$$

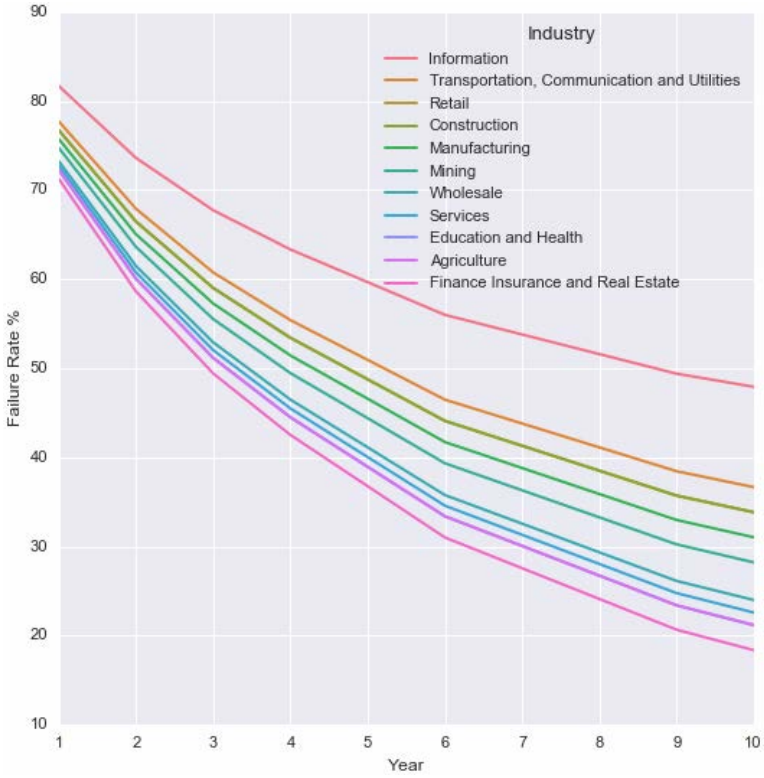


Figura 6.3: Posibilidad de quiebra de una empresa en función de su sector y edad

Y de esta manera ya se podría hacer una estimación del riesgo de una empresa dada su actividad empresarial y su edad. Como se puede observar, el sector con más riesgo es el de tecnologías de la información mientras que el más seguro sería el de finanzas, seguros e inmuebles. Por otro lado, se decide trabajar con SR o *Success Rate* en lugar de con *Failure Rate*, por lo que se utilizará la transformación $SR = 1 - FR$.

Capítulo 7

Flujos de caja libres

Como se explicó en la sección 2.2, uno de los *inputs* más importantes del método de valoración son los FCF, *Free Cash Flows* o flujos libres de caja. La obtención de los mismos es el objetivo de este capítulo. Los flujos de libres de caja se pueden escribir así (se sugiere al lector recordar los acrónimos explicados en la sección 2.2):

$$FCF = OCF - CAPEX$$

$$OCF = Revenues - OPEX$$

No obstante, como se adelantó en la sección 2.3, adaptado a las necesidades del problema quedaría así:

$$FCF = Revenue \times (Ratio_1 - Ratio_2)$$

donde:

$$Ratio_1 = \Big(\frac{OCF}{Revenue}\Big), \qquad Ratio_2 = \Big(\frac{CAPEX}{Revenue}\Big)$$

El problema se dividirá en tres subproblemas que se abordarán en las siguientes secciones:

- **Ingresos anuales** (7.1): Se explicará el proceso de elaboración de un regresor capaz de estimar los ingresos de una empresa.
- **Flujos de caja operativos** (7.2): Se explicará el proceso de elaboración de un regresor capaz de estimar el $Ratio_1$ para poder estimar los OCF a partir de los ingresos estimados en 7.1.
- **CAPEX** (7.3): Si al resultado de 7.2 se le resta el CAPEX se obtendrán los FCF. En lugar de estimar gastos en inversiones de capital, se estimará directamente el $Ratio_2$.

Estos serán entonces los objetivos específicos de éste capítulo, pero no los únicos. Los objetivos generales del proyecto (sección 3.3) siempre tendrán que ser tenidos en cuenta y deberá ayudarse a su cumplimiento en cualquier momento que surja la oportunidad. Por tanto, cada vez que se analiza un dataset se debe extraer el máximo conocimiento posible ya que puede ser de gran utilidad.

7.1. Estimando los ingresos anuales

En esta sección se documentará el proceso de obtención de un modelo capaz de estimar los ingresos de una empresa. Se utilizarán técnicas de Aprendizaje Automático por lo que se aplicarán las fases de la metodología *CRISP-DM*.

7.1.1. Entendimiento del negocio

El objetivo primordial es obtener los ingresos anuales de una empresa ya que se trata de un *input* fundamental en la fórmula del método de valoración (ecuación 2.4 en la sección 2.3). Los requerimientos del problema global juegan un papel importante también aquí. Como no se dispone ni de los ingresos ni de otros indicadores financieros para la mayoría de compañías del dataset objetivo, nace el problema de estimarlos.

Por un lado, aunque en la mayoría de los países es teóricamente legal obtener estos datos (cuentas de resultados), hacerlo no es viable porque supondría tener que ir a las instituciones de cada país en cada caso e ir superando las trabas de las instituciones públicas. Por otro lado, existen organizaciones que realizan inversiones importantes para precisamente recopilar manualmente informes de compañías a través de la administración de distintos países, y como la información suele ser pública, unifican y comercializan los informes. Global Incubator está negociando con algunas organizaciones de este tipo ya que sería una fantástica fuente de información para cumplir con los objetivos de este proyecto y tendría un importante potencial para entrenar modelos muy interesantes. En el momento de realización de esta fase las negociaciones estaban todavía abiertas por lo que no se tuvo acceso a las cuentas de resultados de las empresas.

Contextualizando este subproblema en los objetivos y consideraciones básicas del problema general (sección 1.3), es importante tener en cuenta que existen compañías de muy diversa índole en el dataset objetivo, por lo que la solución adoptada para este problema en concreto deberá ser válida para el mayor número de casos y por supuesto, escalable.

Merece la pena entonces tener el siguiente hecho en cuenta: en el caso de empresas que sí tienen ingresos (o que parece razonable suponerlo así) la validez de la estimación de los mismos no va más allá de la calidad de los resultados que arroje la fase evaluación del modelo. No obstante, ¿qué debería ocurrir con las *startups* que no generen ingresos? A priori el hecho de que una empresa genere o no ingresos sugiere que ambos casos deberían ser tratados de manera diferente a la hora de ser valoradas, y *especialmente* si el método de valoración va a estar fundamentado en el descuento de flujos de caja

En primer lugar podría pensarse en implementar un modelo de Aprendizaje Automático que detectase si una compañía concreta está en una fase en la que genera ingresos o no, para a continuación tratar ambos casos de manera distinta. Aunque, más allá de cómo de fácil/difícil sería conseguir un conjunto de instancias para entrenar dicho modelo, ¿hasta qué punto es necesario realmente hacer esta rigurosa distinción? No se debe olvidar que, especialmente en los primeros pasos de una *startup*, es frecuentemente comentado entre los *Venture Capitals* que 'valorar es un arte y no una ciencia'. En esta tesitura, se propone que el modelo de valoración estime, en los casos en los que la compañía tenga ingresos (desconocidos), los ingresos, y en los casos en los que no los tenga, los que podría llegar a tener. Este último enfoque se puede afinar algo más justificando en primer lugar que el modelo a generar va a ser capaz de medir el potencial de las compañías, cubriendo la principal desventaja de la aplicación tradicional del modelo de descuento de los flujos de caja a *startups* (dificultad para estimar el potencial). Por otro lado, al tratarse así el problema, no sería

necesario diferenciar entre ambos casos y el modelo podría ser aplicado de la misma manera.

7.1.2. Obtención y entendimiento de los datos

Si se quiere aprender a estimar las ganancias de una startup lo primero que se necesitará serán unos cuántos ejemplos de empresas con sus respectivos ingresos en un año concreto. Es muy importante que además se disponga de otra información que en primer lugar muestre algún tipo de correlación con los ingresos y que en segundo lugar se disponga de ella en el dataset sobre el que aplicar posteriormente el modelo o al menos se sepa con certeza que se podrá obtener. La búsqueda de dicha información en internet supone un reto principalmente porque:

- Aunque suele ser pública, es de difícil acceso, de tal manera que es normal encontrarse con fuentes que cobran por este tipo de información.
- Las startups son empresas jóvenes que no suelen generar ingresos en sus primeros pasos.
- Los inversores tienen menos interés en conocer los ingresos de la startup que en saber el crecimiento que experimenta (y experimentará) la compañía. Esto implica que importantes plataformas/bases de datos de startups como *Mattermark*, *AngelList* o *CrunchBase* no inviertan tanto esfuerzo en sacar a la luz los ingresos de sus startups

Scrapeando Inc 5000

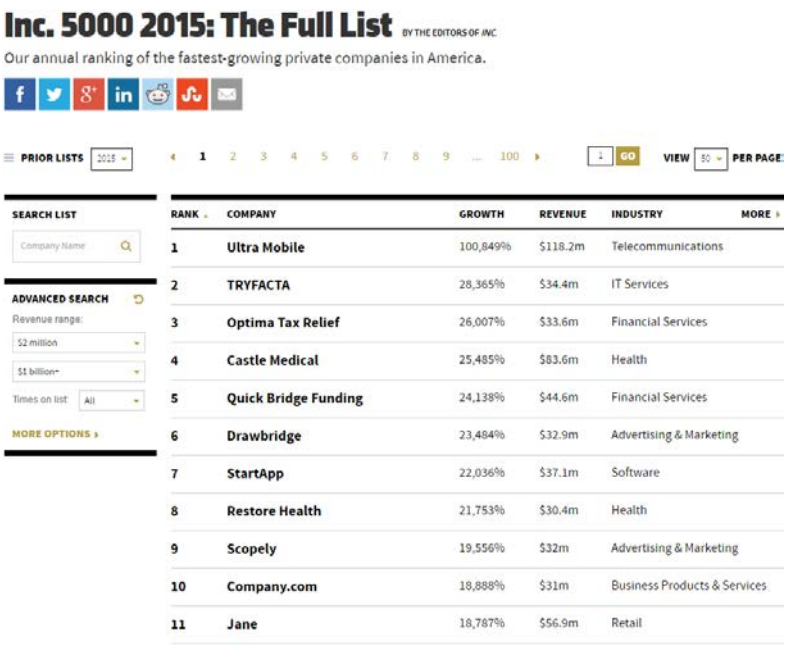
Tras explorar varias posibles fuentes de información a partir de las que construir el dataset de entrenamiento se opta por utilizar los datos de *Inc. 5000*. En ésta página web existen listados que se corresponden con compañías que han experimentado un importante crecimiento en el año X, listándose varios campos de cada una de ellas, entre los que se encuentran los ingresos anuales. En concreto, se tienen listados de entre 2007 y 2015 para empresas que operan en Estados Unidos[1] y de 2015 y 2016 para Europa[2], cada lista tiene entre de 3000-5000 empresas y hay 11 listas en total.

Es importante considerar un aspecto importante que conlleva entrenar con este dataset: al ser un dataset de compañías que han experimentado un crecimiento positivo, se puede deducir que existirá cierto sesgo en la población. Si se entiende como 'positivo' una buena valoración y 'negativo' como mala, esto hará que los verdaderos negativos del modelo de valoración general disminuyan mientras que los falsos positivos posiblemente aumenten algo también. No obstante, las valoraciones relativas no tienen por qué verse afectadas y por lo tanto esto no supone una amenaza al primer objetivo crucial del proyecto (sección 1.3).

Dado que la página web sólo deja exportar directamente los datos de la lista de Estados Unidos en 2015 se optará por hacer *web scrapping* para obtener la información de un número mayor de listas y por ejemplo poder entrenar con más instancias.

Aunque la metodología a utilizar sigue la misma línea que en el caso de extracción del árbol sectorial de `angel.co/markets` (inspección de elementos `HTML` con un navegador e implementación de un *script* que los localice automáticamente), la manera de `inc.com/inc5000` de estructurar la información de interés es bastante distinta, lo que hace que el proceso de obtención e incluso la tecnología a utilizar también lo sean.

[1] www.inc.com/inc5000
[2] www.inc.com/inc5000eu



Inc. 5000 2015: The Full List BY THE EDITORS OF INC

Our annual ranking of the fastest-growing private companies in America.

PRIOR LISTS 2015 | 1 2 3 4 5 6 7 8 9 ... 100 | GO | VIEW 50 PER PAGE

SEARCH LIST — Company Name

ADVANCED SEARCH — Revenue range: $2 million – $1 billion+ — Times on list: All — MORE OPTIONS ›

RANK	COMPANY	GROWTH	REVENUE	INDUSTRY	MORE
1	Ultra Mobile	100,849%	$118.2m	Telecommunications	
2	TRYFACTA	28,365%	$34.4m	IT Services	
3	Optima Tax Relief	26,007%	$33.6m	Financial Services	
4	Castle Medical	25,485%	$83.6m	Health	
5	Quick Bridge Funding	24,138%	$44.6m	Financial Services	
6	Drawbridge	23,484%	$32.9m	Advertising & Marketing	
7	StartApp	22,036%	$37.1m	Software	
8	Restore Health	21,753%	$30.4m	Health	
9	Scopely	19,556%	$32m	Advertising & Marketing	
10	Company.com	18,888%	$31m	Business Products & Services	
11	Jane	18,787%	$56.9m	Retail	

Figura 7.1: Vista general de compañías en una de las listas de `inc.com/inc5000`

Estructura de la información en la web

La principal diferencia es que en el caso del árbol sectorial sólo había que hacer una descarga de un fichero `.html`, por lo que se podía hacer manual sin mayor complicación. No obstante, y como era de esperar, `inc.com/inc5000` no muestra todas las compañías en un mismo `.html`. Por un lado, se dispone de una lista principal con un paginador donde se muestra cada compañía con alguno de sus atributos (figura 7.1). En esta visualización se dispone de bastante información relevante. Hasta hay un botón arriba a la derecha que muestra más columnas. En resumen, se muestra:

- Nombre
- Ingresos
- Industria
- Número de empleados
- Posición en el ranking
- % Crecimiento a 3 años
- Estado (EEUU)
- Ciudad
- Años

Se tiene incluso la opción de cargar un máximo de 200 startups por visualización, por lo que a priori resultaría viable desplegar esas 200 y descargarse los 25 ficheros `.html` que compondrían una lista.

No obstante, es conveniente tener en cuenta que al hacer click en una compañía se muestra una vista individual (ver figura 7.2) de la misma. De la misma manera que ocurre en páginas

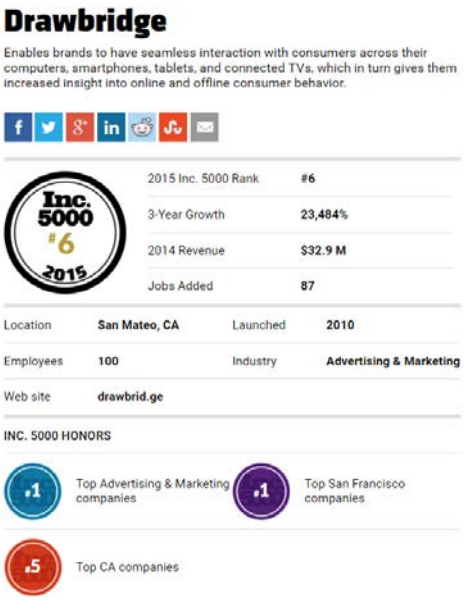


Figura 7.2: Vista individual de una compañía perteneciente a una de las listas de `inc.com/inc5000`

como `twitter.com`, si se hace scroll vertical hacia abajo en esta vista se van cargando el resto de elementos. En este caso, cada elemento sería la vista individual de cada compañía y el orden que seguirían vendría determinado por su ranking (el ordenamiento por defecto de la vista general). La información que se muestra en la vista individual que no aparece en la vista general varía tanto en función de la compañía como en función de la lista en sí:

- No se muestran ciudades para las startups que pertenecen a listas europeas (EU2015 y EU2016).
- Si se dispone de una descripción de la compañía, ésta aparecerá exclusivamente en la vista individual.
- La `url` de la página web sólo aparece en la vista individual.

La descripción puede resultar útil para extraer palabras clave y complementar a la industria en la caracterización de la actividad de la empresa, pero el principal motivo por el cuál merece la pena analizar la vista individual además de la general es que sólo en ella se muestra la página web de la empresa. Este elemento es muy importante ya que permite identificar a una empresa (no con un 100 % de precisión) ofreciendo la posibilidad de hacer *matching* con la información de otros datasets. Esto posibilitaría enriquecer los datos de entrenamiento para en definitiva obtener un mejor modelo predictivo. No sólo posibilitaría completar los valores perdidos de países o número de empleados de algunas compañías, sino que además dejaría la puerta abierta a añadir nuevos atributos como por ejemplo la presencia en redes sociales, algo que perfectamente podría estar correlacionado con los ingresos de una compañía y se hecha en falta en las listas de `inc.com`.

Descargando los datos

Dado que no existe información relevante en la vista general que no salga en la individual, la principal intención será ir a por ésta directamente. Se pueden seguir estrategias distintas para ello. Se asume que sea cuál sea la estrategia, habrá que repetirla manualmente tantas veces como listas se quieran descargar ya que al haber sólo 9 no merecería la pena automatizar el pasar de una lista a otra (pero dentro de cada una sí).

- **Manual**: Descargar todas las vistas individuales de cada compañía manualmente utilizando un navegador web (5000 vistas por lista).
- **Automático**:
  1. Implementar un *scrapper* que vaya haciendo scroll vertical y descargue los ficheros `.html` de cada vista individual haciendo una petición tras otra él mismo.
  2. Implementar un analizador de código `HTML` que extraiga la información de cada vista individual descargada.
- **Híbrido**:
  1. Descargar manualmente las 25 páginas de la vista general.
  2. Implementar un *script* que analice los ficheros `.html` descargados y genere una lista de los links que llevan a cada una de las vistas individuales de las empresas.
  3. Implementar un *scrapper* que utilice la lista anterior para visitar directamente cada una petición a petición por él mismo.
  4. Implementar un analizar de código `HTML` que extraiga la información de cada vista individual descargada.

Respecto a la opción manual, asumiendo que de media se tardarían 6 segundos en descargar una vista individual, el proceso total tardaría:

$$9 \quad listas \times 5000 \quad empresas \times 6 \quad segundos = 3{,}125 \quad d\acute{i}as$$

Resulta mucho tiempo y sería bastante tedioso. Además, no se aprendería absolutamente nada durante el proceso y no sería una solución escalable en absoluto. Queda totalmente descartada.

Respecto a las otras dos opciones, la diferencia estriba en si el coste de automatizar la obtención de los links a los perfiles individuales compensa el tiempo gastado en descargarlos manualmente por tandas de 200.

Antes de nada hay que recalcar que en cualquier caso, para descargar desde *Python* la información de `inc.com/inc5000` no basta con hacer peticiones web a la `url` correspondiente utilizando librerías básicas como *urllib*[3] o *requests*[4]. Ninguno de los métodos de *BeautifulSoup* será capaz de encontrar los elementos web que queremos básicamente porque no se encontrarán en el fichero *.html* descargado. Por tanto se utilizará *Selenium*, ya que como se explicó en la sección 4.1.2, permite simular el uso humano de un navegador de internet y entre otras cosas, ejecutará el código *Javascript* de la página necesario para mostrar su contenido.

Volviendo al tema de las estrategias, se ha optado por la híbrida. El motivo a continuación:

[3] `docs.python.org/3/library/urllib.html`
[4] `docs.python-requests.org/en/master/`

Código 7.1: Espera inicial en Selenium

```
with closing(Chrome(driver_path)) as browser:
    browser.get(url)
    page_source = browser.page_source
```

Se estima que la parte manual de esta opción consumirá:

$$9 \quad listas \times 25 \quad páginas \times 6 \quad segundos = 22{,}5 \quad minutos$$

Que sólo habría que hacerlo una vez. Por otro lado, no sería necesario interactuar demasiado con el código *Javascript* de la página, ya que los fragmentos de código necesarios para mostrar todo el contenido de la vista individual de una compañía se ejecutan automáticamente al principio, así que simplemente habría que dar unos segundos para que se ejecute.

El código mostrado en 7.1 muestra cómo de sencillo resulta descargar el código `HTML` a partir de una `url` dando un margen de 5 segundos a que se ejecute todo lo que se tenga que ejecutar (código *Javascript*) antes de guardar en memoria el contenido completo. La variable final *page_source* tendría almacenado todo el código listo para analizar con *BeautifulSoup*. La elección de la estrategia 100 % automática ahorraría tiempo pero implicaría interactuar más activamente con la página web ya que cada vez que se descargase un perfil habría que hacer *scroll*. No tiene porque ser difícil, pero posiblemente hubiese que invertir más de 22,5 minutos en implementación y no se sabe con seguridad cuál sería el comportamiento del sistema cuando llevase muchas peticiones acumuladas, ya que cada vez que se hace *scroll* se añaden más datos a la página y es posible que ello conllevase algún problema de memoria.

En definitiva, aunque quizás la estrategia automática sea más elegante es menos apropiada en esta situación. El factor negativo más importante de renunciar a ella es no adquirir la experiencia que implicaría realizar un *web scrapping* más avanzado, pues el dominio de este campo resulta de gran utilidad hoy en día.

Cada fichero `.html` descargado manualmente se corresponde con una página de la vista general (figura 7.1) de una lista, conteniendo información de 200 compañías. Como toda la información que aparece en esta vista también aparecerá en la vista individual (figura 7.2) lo único que hay que extraer del código `HTML` son los 200 enlaces a los perfiles individuales de cada compañía.

Como era de esperar, la información dentro del código `HTML` está muy bien estructurada. La imagen 7.3 es prueba de ello. En ella se ve que los enlaces al perfil individual de cada compañía están situados dentro de elementos fácilmente diferenciables del resto.

Código 7.2: Descarga de enlaces a perfiles de compañías de `inc.com/inc5000`

```
def downloadIncList(html_path)
        page_source=open(html_path).read()
        soup = BeautifulSoup(page_source)
        for url_ in soup.find_all('td','c2'):
                url=url_.find('a',target='_blank').get('href')
        company=downloadCompanyProfile(url)
        with open(file_path,'w') as fp:
                        json.dump(company,fp)
```

Figura 7.3: Código de la vista general de una lista de `inc.com/inc5000`

Así, una simplificación del código encargado de extraer los enlaces se muestra en el código 7.2. El código real resulta algo más complejo ya que se controla cualquier excepción que pueda originar la función *downloadCompanyProfile()*, otorgando un máximo de 3 intentos a cada petición, incrementando los tiempos de espera en cada una. Sólo se descargará un perfil si éste no existe en disco ya, lo que permite poder parar y reanudar las descargas cómodamente.

La extracción de los enlaces resulta sencilla gracias al método *find_all()* de *BeautifulSoup*, que devuelve un listado con todos los elementos `HTML` que pasen por el filtro definido. Hecho esto, iterar sobre el listado recogiendo el enlace que contiene cada elemento resulta trivial. Posteriormente se escribe en disco directamente un archivo exclusivo para cada compañía donde su información queda estructurada como un objeto *JSON*(una lista de elementos $< clave >, < valor >$).

Cada enlace extraído se le pasa a la función *downloadCompanyProfile(url)*, cuya parte inicial, encargada de descargar el fichero `.html` se mostró en el listado 7.2 y se muestra completa (pero simplificada) en 7.3. Esta función debe analizar el documento `HTML` de una empresa y extraer exclusivamente la información deseada. En primer lugar, hay que tener cuidado a la hora de implementar la función ya que no todos los perfiles de compañías contienen los mismos campos. Por ello, la estructura de datos utilizada para almacenar cada compañía será un diccionario (se podría decir que es el equivalente a un objeto *JSON* en *Python*), ya que permite ir añadiendo

Código 7.3: Descarga de un perfil de una empresa de `inc.com/inc5000`

```
def downloadCompanyProfile(url)):
    with closing(Chrome(driver_path)) as browser:
        browser.get(url)
        page_source = browser.page_source
        soup = BeautifulSoup(page_source)
        company=dict()
        company['inc_url']=url
        header=soup.find('header')
        company['name']=header.find('h1').string
        company['description']=header.find('p').string
        info=soup.find_all('dl')
        for i in info[:2]:
            names=[item.string for item in i.find_all('dt')]
            values=[item.string for item in i.find_all('dd')]
            for key,value in zip(names,values):
                company[key]=value
        return company
```

claves nuevas según vayan apareciendo. No obstante, hay algo que sí que se mantiene constante en todos los perfiles de las empresas: el nombre y la descripción (si la hay) siempre vienen en un elemento *header* (ver figura 7.4). Por lo tanto, dicha parte se ha codificado explícitamente en la función del listado 7.3.

```
▼<header>
   <h1>Ultra Mobile</h1>
  ▼<p>
     "A mobile virtual network operator offering various mobile plans
     that feature international talk, text, and data services at no
     additional charge."
   </p>
  ▶<div class="ShareContainer" id="share36631">…</div>
 </header>
```

Figura 7.4: Código del header de una vista individual en una lista de `inc.com/inc5000`

Como sugiere la figura 7.2, la información de cada compañía está estructurada en bloques de información independientemente del *header*. Dichos bloques se corresponden con elementos `HTML` de tipo *dl*. De todos los que hay, sólo nos interesan los dos primeros. Ya que por ejemplo, en la figura 7.2 se muestra un tercer bloque con honores otorgados a la compañía por la propia web. Probablemente, esta información está correlacionada con los ingresos de la empresa, pero no tendría sentido incluirla en el entrenamiento del modelo si el objetivo es aplicarlo a empresas que no estén en él. Por tanto, la información de ese bloque no se recogerá y tampoco la de los siguientes, que no se muestran en la figura 7.2 pero al contener información del estilo 'ranking en listas anteriores' se les aplica el mismo razonamiento. Respecto los dos primeros bloques *dl* (los que sí son de interés), la información que contienen varía según el caso, como expone la figura 7.5. Por suerte, independientemente del número de campos y del nombre de los mismos contenidos en cada bloque, siempre se cumple que el nombre del campo está dentro de un elemento *dt* y su valor de un *dd* que además, siempre viene

detrás. Éste hecho, junto con la flexible sintaxis de *Python*, permiten abarcar todos los casos de una manera elegante (ver últimas líneas del listado 7.3).

Se puede concluir que la totalidad de los datos necesarios para entrenar el modelo que cumpla los objetivos especificados ha finalizado con éxito. En la metodología *CRISP-DM*, antes de pasar a la fase de preparación de los datos es habitual realizar un entendimiento de los datos mediante análisis estadísticos que arrojen algo de conocimiento acerca del dataset a construir. No obstante, dado que que los datos todavía no se encuentran en un estado manejable y requieren una importante inversión en unificación y preparación, será después de la siguiente fase cuando los datos se encuentren en un formato más amigable para efectivamente poder aplicar un análisis de datos que ofrezca una visión más rica. Por tanto, se pospone dicho estudio a una segunda iteración (posterior a la preparación de los datos completa).

```
▼<section class="panel panel-badge">
  ▼<div class="content">
     <img class="badge" src="http://images.inc.com/img gen/
     inc5000 2015 europe/inc5000 num 15.png">
    ▼<dl>
       <dt class="2015 inc. 5000 rank">2015 Inc. 5000 Rank</dt>
       <dd class="2015 inc. 5000 rank">#15</dd>
       <dt class="growth">3-Year Growth</dt>
       <dd class="growth">3,376%</dd>
       <dt class="2013 revenue">2013 Revenue</dt>
       <dd class="2013 revenue">€7.4 M</dd>
       <dt class="jobs added">Jobs Added</dt>
       <dd class="jobs added">829</dd>
     </dl>
   </div>
  </section>
  <!--Inc5000 only-->
  <!--Inc5000 only-->
▼<section class="panel panel-address">
  ▼<dl>
     <dt class="country">Country</dt>
     <dd class="country">Romania</dd>
     <dt class="employees">Employees</dt>
     <dd class="employees">832</dd>
     <dt class="industry">Industry</dt>
     <dd class="industry">Consumer Products & Services</dd>
     <dt class="website">Web site</dt>
   ▶<dd class="website">…</dd>
    </dl>
  </section>
```

```
▼<section class="panel panel-badge">
  ▼<div class="content">
     <img class="badge" src="http://images.inc.com/img gen/
     inc5000 2013/inc5000 num 268.png">
    ▼<dl>
       <dt class="2013 inc. 5000 rank">2013 Inc. 5000 Rank</dt>
       <dd class="2013 inc. 5000 rank">#268</dd>
       <dt class="growth">3-Year Growth</dt>
       <dd class="growth">1,603%</dd>
       <dt class="2012 revenue">2012 Revenue</dt>
       <dd class="2012 revenue">$8.9 M</dd>
       <dt class="jobs added">Jobs Added</dt>
       <dd class="jobs added">80</dd>
     </dl>
   </div>
  </section>
  <!--Inc5000 only-->
  <!--Inc5000 only-->
▼<section class="panel panel-address">
  ▼<dl>
     <dt class="location">Location</dt>
     <dd class="location">Herndon, VA</dd>
     <dt class="launched">Launched</dt>
     <dd class="launched">2004</dd>
     <dt class="employees">Employees</dt>
     <dd class="employees">100</dd>
     <dt class="industry">Industry</dt>
     <dd class="industry">IT Services</dd>
     <dt class="website">Web site</dt>
   ▶<dd class="website">…</dd>
    </dl>
  </section>
```

Figura 7.5: Código de dos perfiles de empresas distintas, con campos distintos pertenecientes a una lista de `inc.com/inc5000`

7.1.3. Preparación de los datos

En *Data Science*, una de las partes más importantes y en las que a menudo hay que invertir más tiempo es en organizar, limpiar e incluso corregir los datos con los que se trabajará posteriormente. En este apartado se cubre la fase inicial de este proceso.

Unificando los datos

Los datos recogidos de *inc5000* contienen información incompleta. Aunque afortunadamente se dispone de los ingresos anuales de todas las compañías, no se tiene el país ni la industria (entre otros) de todos, y se presupone que son factores que de una manera u otra afectarán a los resultados financieros. Para poder entrenar posteriormente el modelo no deberían aparecer valores nulos en las instancias, por lo que una opción sería rellenar los campos que falten, si no faltan muchos, con los valores más frecuentes (la moda). No obstante, ya que se dispone de otra base de datos con información de millones de compañías, lo conveniente sería mirar a ver si los datos que faltan se encuentran allí.

Ahora bien, habrá que definir una estrategia para hacer el *matching*. Como se explicó anteriormente, una buena idea sería utilizar el campo de la página web de la empresa para hacerlo ya que es el campo más cercano a ser único en cada base de datos. Un rápido vistazo a dicho campo en ambas bases de datos basta para darse cuenta de un problema que habrá que solucionar antes de poner a prueba con millones de datos la potente función *merge* de *Pandas*, y es que las `urls` aparecen con distintos formatos y variaciones:

- Algunas empiezan por `https://` o `http://` y otras no tienen las `//`.
- Algunas tienen puesto `www.` y otras no.
- Algunas incluyen información como `/en-us/`, `/home/`, `/page/` y otras no.
- Algunas contienen extensiones `.php`, `.asp`, `.aspx` o `.html`.
- Algunas incluyen directamente la página de sus redes sociales.
- Algunas incluyen información no útil como `//es/?ir=1`, `&amp`, `#` etc.

Con tantas variaciones una cosa es seguro, si se intenta hacer un *matching* que mire por el valor exacto va a haber muchas compañías que se pierdan por el camino. Se debe de elegir entonces una estrategia para asegurar un *matching* que devuelva un alto porcentaje de *matches* y en un tiempo razonable. Se proponen dos maneras de abordar el problema (y una tercera mixta):

- **Búsqueda borrosa**: Se podría hacer el *matching* de tal manera que un *match* pueda ocurrir no sólo cuando dos valores sean exactamente iguales, si no cuando sean muy parecidos. De esta manera, no se perderían aquellas `urls` que se diferencian por unos pocos caracteres. Esto se podría conseguir mediante la Distancia de *Levenshtein*, que mide el número mínimo de modificaciones a nivel de un carácter que habría que efectuar para que ambas palabras acabasen siendo iguales [Nav01]. Además, ya existe una implementación en *Python* llamada *FuzzyWuzzy*. En su documentación explica que se incluyen 4 versiones de dicha distancia que se adaptan mejor a cada tipo de problema, cada una con sus particularidades [11]. Por ejemplo, existen versiones en las que el orden de los caracteres o subcadenas se penaliza más que otras. Tras estudiar las 4 en su documentación, se concluye que la que mejor se adapta al problema presente es una que antes de calcular la distancia de *Levenshtein* divide el *string* en *tokens* deshaciéndose de los puntos, barras etc. La ventaja de esta estrategia sería que si se ajusta correctamente el umbral se podrían obtener bastantes verdaderos positivos. Por contra, no se podría utilizar la función *merge* de *Pandas* y, a pesar de que la librería ofrece una clase *Process* para el cálculo de distancias por lotes, es infinitamente más lenta que aquella.

- **Limpiar urls**: Esta estrategia consistiría en limpiar la url hasta quedarse sólo con la parte que identifica a la empresa, para luego hacer el *matching* normal con las `urls` limpias. Para que la limpieza sea lo más efectiva posible, se podría utilizar una combinación de reemplazos y expresiones regulares en *Python*. Su principal ventaja es poder utilizar el *merge* de *Pandas* y su contra que existan algunos falsos negativos.
- **Híbrido** Básicamente, limpiar las `urls` primero y luego realizar un *matching* borroso. Su ventaja es que se maximizaría el *recall* (se recuperarían muchísimos verdaderos positivos) pero a un precio muy alto, ya que no sólo se renunciaría al *merge* de *Pandas* sino que al estar la `url` limpia, habría que subir mucho el umbral de similitud, haciendo difícil el contraste.

Dado que dos compañías pueden ser muy similares en el nombre, diferenciándose solamente por un carácter, se descarta la estrategia híbrida. Por otro lado, la información de la `url` que no es el nombre o el 'núcleo de información' es muy variable y no aporta demasiada relevancia, por lo que no sólo sería complicado establecer un umbral fijo óptimo sino que además se estaría teniendo en cuenta a la hora de medir la similitud información que realmente no influye (quizás el dominio de la página sí). Además, en el caso de la búsqueda borrosa, habría que efectuar

$$40000 \times 8000000 = 320 \quad millones \quad de \quad medidas$$

No se ha calculado en unidades de tiempo ya que para hacerlo bien habría que tener en cuenta lo que se tarde en referenciar los datos. De todas formas, con los argumentos anteriormente dados ya es suficiente para trabajar en una función que limpie las `urls`.

El desarrollo de dicha función no ha sido complicado. La manera de proceder ha sido: primero implementar la limpieza de aquellos elementos más básicos listados en el apartado anterior e ir añadiendo funcionalidades, lo más genéricas posibles según se vayan detectando `urls` con distintos tipos de 'basura' después de probar lo que ya se tiene desarrollado. Ya sean reemplazos o expresiones regulares, todas las decisiones y reglas siguen el mismo principio: eliminar toda la información que no aporte identidad a la empresa. De entre todas las decisiones y reglas realizadas, la más determinante ha sido la del tratamiento de la barra separadora `/` que suele dejar a la izquierda el nombre principal de la página y el dominio de nivel superior (`.com`, `.co.uk`, `.net` etc.) y a la derecha información no útil la mayoría de los casos. No obstante, algunas empresas (sobre todo las grandes) aparecen en las bases de datos tantas veces como divisiones tengan. Así, la *Compañía X* dedicada a la tecnología y entretenimiento puede dividirse y aparecer dos veces en la base de datos, primero como `www.Xcompany/mobile` y otra `www.Xcompany/tv`. Si se optase por eliminar desde la barra hacia la derecha se acabaría con dos instancias con la misma `url`. Esto es problemático, en primer lugar, estamos hablando de un campo que se utiliza para identificar filas y no debería perder esta propiedad. En segundo lugar, hay que tener mucho cuidado con los duplicados a la hora de hacer *merges*, sobretodo si se va realizar un *full-outer join*. En el caso que nos incumbe, como lo que queremos es pasar los datos de una base de datos a otra, habría que realizar un *left-outter join* (o *right-outter join*, según se mire), por lo que bastaría con asegurarse de que la base de datos que contiene la información que queremos incluir en la de *inc500* no tiene duplicados para no acabar con millones de filas, pudiendo dejar algunos duplicados en la de *inc500* sin que ello causase demasiado problema. Ahora bien, ¿bajo qué criterios se debería borrar el duplicado de la *Compañía X*? No hay una solución que sea la mejor respuesta en todas las situaciones posibles, por lo que habrá que intuir cuál es la más óptima. Éstas son algunas de las ideas que se barajan:

- Quedarse con la que tenga más empleados.

- Quedarse con la que tenga más seguidores en redes sociales.
- Quedarse con la que tenga menos valores nulos.
- Juntar todos los atributos en una sola fila

Se eligió la cuarta opción. De esta manera, no se perderá información. En el caso de los empleados y los seguidores en redes sociales se pueden sumar ambos valores, respecto a los campos categóricos como países, ciudades, industria etc. se juntarán en una lista. Esta manera de operar, aunque minimiza la pérdida de información, hace que la base de datos sea menos manejable, pero eso no implica que no lo siga siendo, y eso es lo importante. Además, en *Python* resulta más sencillo de lo que parece utilizar una tabla estructurada de esta manera. Los siguientes objetos y funciones de la librería *Pandas* han resultado extremadamente útiles para todo el proyecto en general, pero en operaciones como esta es donde más provecho se les ha sacado:

- **pandas.DataFrame**: Objeto que representa una tabla, con sus filas, columnas e índices.
- **pandas.Series**: Se podría entender como una lista de valores con índices (como un diccionario de *Python*). Representa un columna de un *DataFrame*.
- **pandas.Series.map(f)**: Ejecuta la la función *f()* sobre los elementos de un objeto *Series*.
- **pandas.DataFrame.apply(f)**: Ejecuta la función *f()* sobre los elementos de un objeto *DataFrame*.
- **pandas.DataFrame.groupby(k)**: Crea agrupaciones de las filas de un objeto DataFrame (al estilo *SQL*) identificando cada grupo por el campo *k*.
- **pandas.merge()**: Ejecuta una operación *merge* (al estilo *SQL*) sobre dos DataFrame.

Desde luego, se han descrito estos elementos de manera superficial para ilustrar de alguna manera cuál es la forma en la que se trabaja (y se ha trabajado) con esta librería. Las funciones y objetos mencionados cuentan con una amplia diversidad de parámetros que se puede consultar en la documentación que, dicho sea de paso, es muy clara y explicativa. En todos los casos, el uso de estas funciones, que están implementadas a bajo nivel, resulta mucho más eficiente que iterar manualmente por las filas de un objeto y realizando las modificaciones pertinentes, ya que entre otras cosas, está última forma implica la creación de objetos intermedios.

A continuación se describe brevemente cómo, con estas herramientas se pasaron finalmente los datos de la base de datos origen a la de *inc5000*:

1. Implementar *cleanURL()*, que recibe una url 'sucia' y la devuelve limpia.
2. Efectuar limpieza mapeando la columna url de la base de datos origen. (*df['url'].map(cleanURL)*)
3. Agrupar las empresas de la base de datos origen utilizando sus urls limpias (*df.groupby('url')*).
4. Implementar la función *customGB()* que recibe una agrupación de filas y la fusiona en una sóla fila (sumando o creando listas para cada campo según corresponda).
5. Creación de la tabla sin duplicados y con información fusionada. (*groupby_object.apply(customGB)*)
6. Efectuar limpieza mapeando la columna url de la base de datos *inc500*. (*di['url'].map(cleanURL)*)

7. Realizar el *matching* entre las dos tablas (*pd.merge(di,df,on='url',how='left')*, equivalente a un *left outer join* de *SQL*).

De las 40000 instancias en *inc5000*, se encontraron *matches* para el 57 %, lo que implica entre otras cosas que si se quiere entrenar el modelo con atributos de la otra base de datos, no se podrán utilizar todas las empresas de *inc5000*.

Preparando los atributos

Ya se ha obtenido una sola tabla a raíz de las dos, pero todavía faltan decisiones por tomar y trabajo por hacer para dejar listo el dataset. Concretamente, hay que decidir cuáles van a ser los atributos finales y ver según el caso como construirlos a partir de las dos fuentes de información de las que se dispone. En todos los casos, las funciones de *Pandas* listadas anteriormente han sido fundamentales para esta parte también:

- **Revenue** (numérica): Sacado de *inc5000*, representa los ingresos anuales de la empresa.
- **Revenue Year** (numérica): Sacado de *inc5000*, indica el año al que pertenecen los ingresos de la empresa.
- **Employees** (numérica): Sacado de *inc5000* y completando los valores nulos con la base de datos origen, representa el número de empleados de la empresa.
- **Followers** (numérica): Sacado de la base de datos origen, representa los seguidores de la empresa en la red social *Linkedin*.
- **Age** (numérica): Sacado de *inc5000* y completando los valores nulos con la base de datos origen, representa la edad en años de la empresa.
- **Domain** (categórica multi-etiqueta): Sacado de *inc5000* y la base de datos origen conjuntamente. Representa el dominio de nivel superior de la página web de la empresa (`.com`, `.co.uk`, `.net` etc.). Es multi-etiqueta porque algunas empresas tienen varias páginas web.
- **Tags** (categórica multi-etiqueta): Sacado de *inc5000* y la base de datos origen conjuntamente. Representa el tipo de actividad de la empresa. Consiste en un conjunto de palabras extraídas con el sistema desarrollado en 5 aplicado sobre la concatenación de los campos de nombre, descripción, industria, palabras clave de ambas bases de datos.
- **GTags** (categórica multi-etiqueta): Sacado de *inc5000* y la base de datos origen conjuntamente. Representan localizaciones geográficas relacionadas con la empresa. Consiste en la unificación del país ciudad, localidad etc. de ambas bases de datos. Por supuesto, no es aceptable que existiese una etiqueta *'Slovakia'* y otra *'Slovac Republic'*. Afortunadamente, la manera de nombrar las localizaciones de ambas fuentes es prácticamente idéntica por lo que el preprocesado requerido para fusionar los datos adecuadamente no ha sido muy trabajoso (en total son 7 reglas de reemplazo).

La idea de las abstracciones *Tags* y *GTags* viene motivada porque es posible que no se cuente con el sector, industria y palabras clave de todas las empresas, por lo que si se decidiese trabajar con dichos atributos como columnas separadas habría que tratar problemas de valores nulos, y de esta manera dichos problemas prácticamente desaparecen. Lo mismo aplica para el país, ciudad,

localidad y estado. Además, en el caso de los atributos multi-etiqueta, cuando se entrene el modelo, cada posible *tag* será codificado con una columna binaria y al algoritmo le dará igual si esa etiqueta es en realidad un pueblo, un dominio o un sector.

El atributo más interesante de la base de datos origen es el de los seguidores en la red social *Linkedin*. Es importarse detenerse aquí a considerar una cosa: la información de los seguidores es de los últimos meses, mientras que los ingresos que aparecen en *inc5000* no tienen por qué ser de este año. Si la *Compañia X* tuvo en 2010 2 millones de € ingresos y así aparece almacenado en *inc5000* pero se ha hecho *match* con la otra base de datos, que tiene información 5 años más reciente, los seguidores de *Linkedin* que se muestren seguramente sean distintos a los que tenía la empresa en 2010. Desde una perspectiva informativa desde luego no tendría demasiado sentido guardar los datos así, sin más. Ahora bien, desde el punto de vista del Aprendizaje Automático, ¿se debería descartar entrenar con dicha instancia? ¿se deberían utilizar sólo empresas cuyos ingresos anuales fueron del último año para poder utilizar el atributo de seguidores en redes sociales? Se analizará en el próximo apartado.

7.1.4. Entendiendo los datos

Antes de pasar a la fase de modelado, y aprovechando que ya se tienen los datos preparados, se procede a realizar un análisis estadístico de los datos.

Análisis univariante del dataset

A continuación se analizan algunos aspectos generales del dataset que se utilizará para abordar el problema. En primer lugar, dado que el interés de este proyecto no es el de valorar compañías grandes, se dejarán fuera del dataset aquellas empresas que tengan ingresos anuales de más de 100 millones de € o más de 500 empleados. Como no aportarían un valor añadido, es mejor dejarlas fuera, ya que además son claros datos atípicos (*outliers*) y algunos algoritmos de Aprendizaje Automático son sensibles a ellos, pudiendo minar el rendimiento predictivo de las compañías objetivo. Tras aplicar este filtro, el dataset se reduce un 10 % aproximadamente, quedando un total de 37498 compañías.

Ahora sí, con estos datos se procede a realizar un análisis estadístico preliminar de sus atributos y output:

Descripción estadística de atributos numéricos							
Nombre	Media	Desv. típica	Perc. 25	Perc. 50	Perc. 75	Mín	Máx
Revenue	15.96	18.56	4.7	9.1	21.1	0.52	99.6
Employees	77.4	89.3	20	45	98	1	499
Followers	1197	4747	94	428	1221	0	416968
Age	10.2	11	6	10.3	16.9	0	220

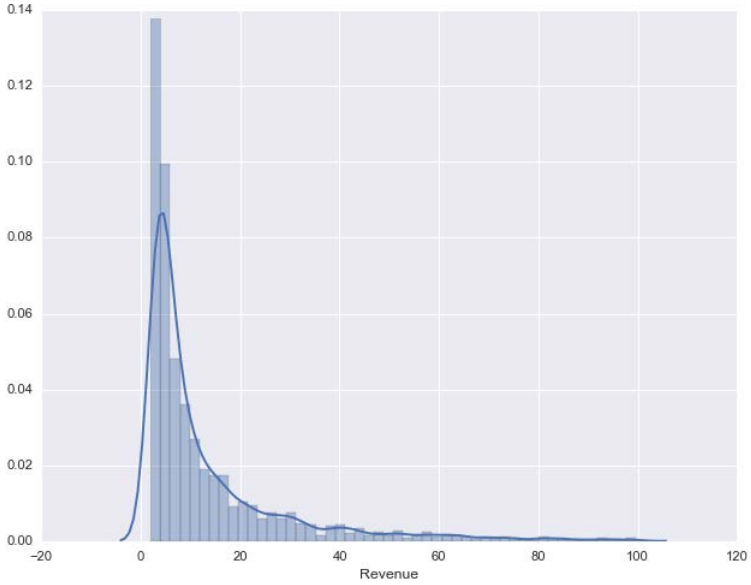


Figura 7.6: Distribución del nivel de ingresos anuales en el dataset

Descripción estadística de atributos categóricos multi-etiqueta					
Nombre	Moda	Frec. moda	Cantidad media	Cantidad máxima	Tipos
Tags	Information technology	22601	15	89	1459
GTags	United States	15893	2	21	4392
Domains	.com	17990	1	3	94

Por otro lado, resulta de especial interés observar cuál es la distribución de los valores del output, *Revenue*. Justo lo que muestra en la figura 7.6. En ella aparece la distribución univariada de los ingresos anuales de las empresas que cumplen el filtrado preliminar. Con tal fin, se ha dibujado un histograma, que entre otras cosas muestra que la gran mayoría de empresas del dataset ingresan menos de 20 millones. Aunque esta información ya aparecía en la tabla (Perc. 75 % es 21.1), la visualización facilita más el análisis. Por otro lado, en la imagen 7.6 también se ha efectuado una estimación de la densidad del kernel (*KDE*), que consiste en inferir cuál es la función de densidad de probabilidad de una variable aleatoria (que en este caso es *Revenue*) a partir de una muestra [Ros56]. Dicha estimación muestra que claramente no se trata de una distribución normal, sino que se parece a más a una distribución F. En cualquier caso, el éxito de un algoritmo de Aprendizaje Automático suele venir bastante antes determinado por las relaciones entre las variables del modelo y el *output* que por la distribución de éste último. Esto ocurre en general, pero en casos muy concretos hay que analizarlo con precaución, como cuando la salida sigue una distribución de *Cauchy*, que no cumple el Teorema Central Del Límite y que tiene propiedades muy peculiares, como colas pesadas y ausencia

de media y varianza, lo que influye directamente en la capacidad de un algoritmo de predecir los valores centrales. Afortunadamente, la figura 7.6 no se parece a una *Cauchy*. Por otro lado, resulta conveniente aclarar que, como es habitual en Aprendizaje Automático, se presupone que el output de las futuras instancias a predecir mantendrá la misma distribución que la de las instancias del entrenamiento (figura 7.6). Esta suposición es muy importante y cuantos más ejemplos se tengan para entrenar, más probabilidades habrá de que se cumpla.

Por otro lado, también podemos analizar visualmente cuál es la frecuencia y distribución de los valores de los atributos. En esta sección se muestra la de *Tags* y *Employees* (figuras 7.8 y 7.7 respectivamente), mientras que las visualizaciones para *Age*, *GTags*, *Domains* y *Followers* se han dejado en el anexo A. La información que se ha mostrado ya es suficiente para ir sacando algunas conclusiones de los datos con los que se va a trabajar:

- De todas ellas se puede decir que los valores están distribuídos de manera irregular y asimétrica.

- Respecto a *Tags* destacan claramente aquellas actividades relacionadas con la tecnología entre las 1459 existentes. En promedio, se ha capturado la actividad de cada empresa con 15 etiquetas. Es un número aceptable, teniendo en cuenta que se incluyen etiquetas amplias que definen sectores y palabras clave específicas.

- Más contraste existe todavía entre el uso de *.com* (presente en el 48 % de las `urls`) y *.net* (el segundo más frecuente) presente sólo en el 2 % de las empresas, existiendo 94 dominios de nivel superior diferentes.

- Del número de empleados se puede decir que hay bastantes empresas pequeñas de las que el modelo va a poder aprender (las empresas pequeñas y medianas son las más interesantes para este proyecto) pero también hay varias empresas grandes, de hecho si se miran los percentiles se deduce fácilmente que el aproximadamente 25 % de las compañías tienen más de 100 trabajadores en nómina.

- La conclusión del tamaño de las empresas está muy relacionada con lo que se puede decir de la edad, cuya distribución es similar (las colas algo más ligeras) y denota que sólo el 25 % de las empresas tienen menos de 6 años, girando la mayoría entorno a los 10. Es un número ciertamente alto para que una empresa pueda ser considerada *startup*, pero eso no implica que el modelo que se construya sobre estos datos no vaya a tener nada que decir sobre compañías más jóvenes. Habrá que construir un modelo capaz de entender las relaciones que existan entre las variables y que pueda generalizar sobre empresas totalmente nuevas, de 3 o 20 años de edad.

- Finalmente, de *Followers* se puede decir que es muy irregular, con valores muy dispersos. De hecho, la visualización de *Followers* colocada en el anexo a tenido que se retocada de tal manera que sólo aparezcan aquellas compañías con menos de 10000 seguidores, ya que aunque el percentil del 75 % sea de 1221, el máximo es de 416968 lo que hace que una visualización de la distribución completa desvirtúe los detalles importantes.

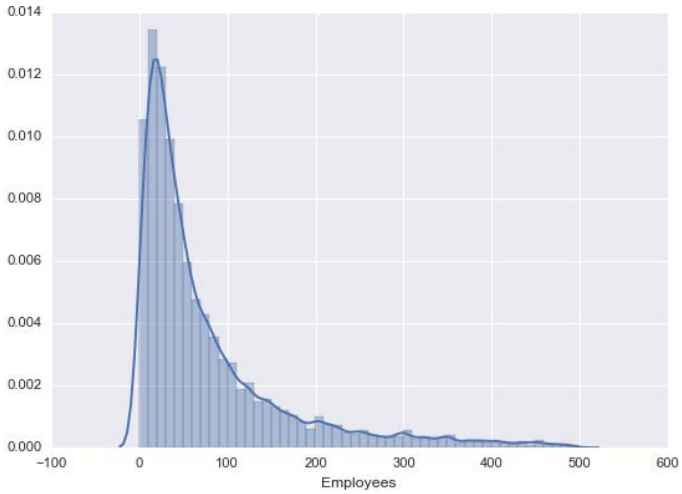


Figura 7.7: Distribución de *Employees* en el dataset

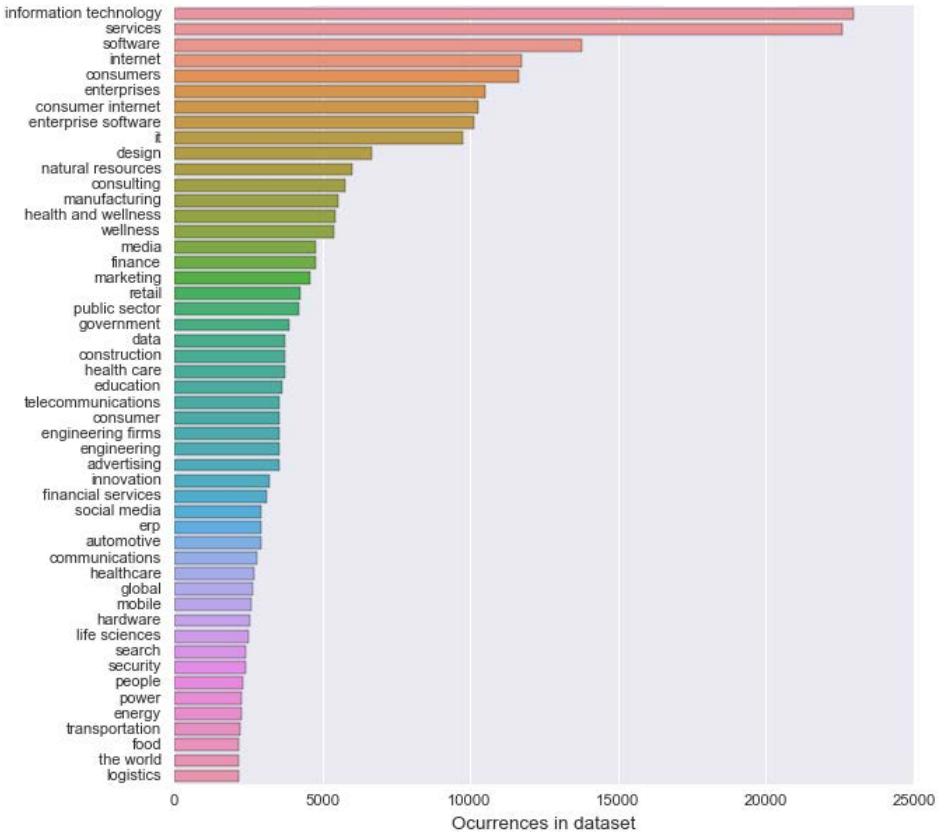


Figura 7.8: Tags más frecuentes en el dataset de entre 1459 existentes

Análisis multivariante del dataset

En esta subsección se elaborará un análisis multivariante de los datos. El objetivo principal será hacerse una primera idea del tipo de relación que existe entre las variables. Es decir, si existe, con qué fuerza y si muestran linealidad o no.

Primero se abordarán los atributos numéricos. En concreto *Age* y *Employees* respecto a *Revenue*. La figura 7.9 expresa visualmente cuál es la relación que existe entre estas variables. Estas son las conclusiones que se extraen de su análisis:

- No parece que exista una relación entre la edad de una empresa y sus ingresos. De hecho se observa, aunque no de una manera excesivamente clara, que hay varias empresas con más de 95 millones de € de ingresos y menos de 10 años de edad. Esta apreciación podría estar muy relacionada con el hecho de que las empresas de *inc5000* se caracterizan por haber experimentado un alto crecimiento. No es algo bueno, ya que indica que la muestra está algo sesgada.
- Como era de esperar, sí existe una correlación positiva entre *Employees* y *Revenue*, ya que más empleados significa tener más recursos productivos, que suele resultar en un mayor nivel

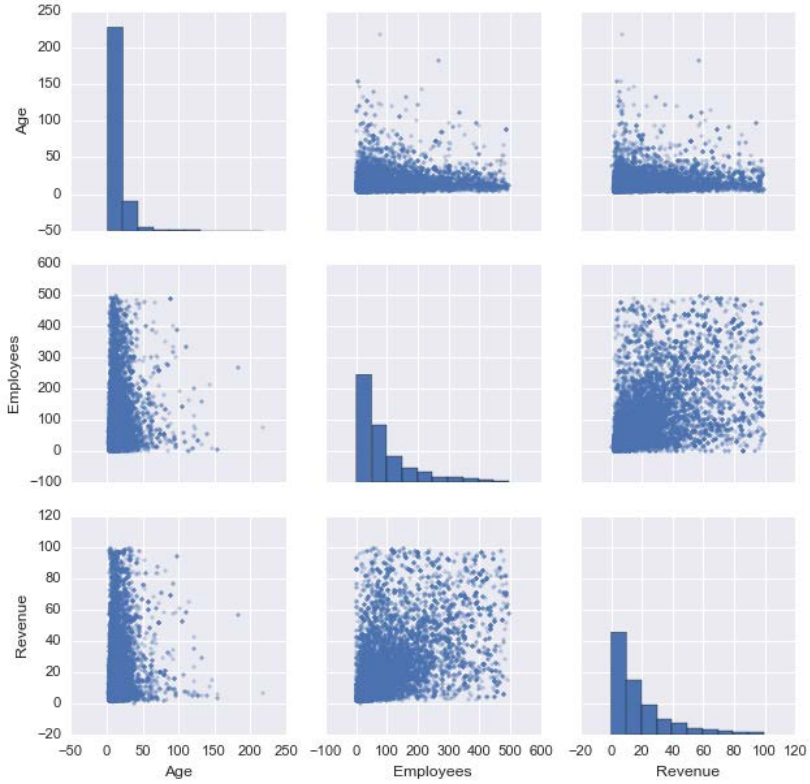


Figura 7.9: *Pair plot* de *Employees*, *Age* y *Revenue*

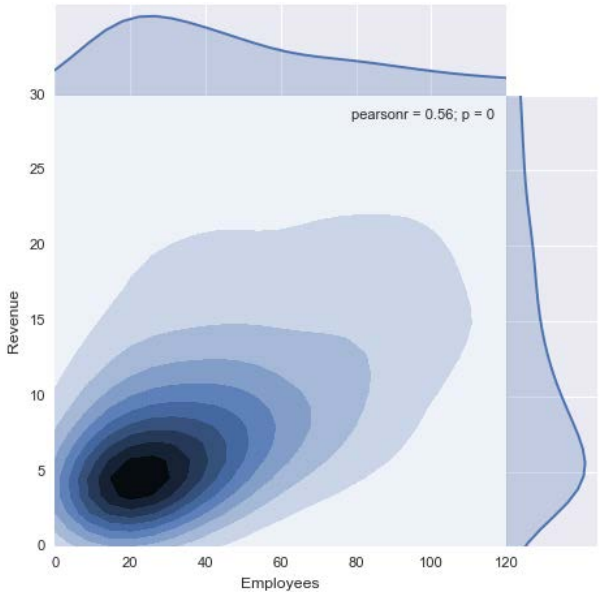


Figura 7.10: *Joint plot* con estimación de la densidad del *kernel* de *Employees* respecto a *Revenue*

ingresos. Tampoco es excesivamente clara en la figura 7.9, pero entre el ruido se puede apreciar cierta relación lineal. Para complementar el análisis de esta variable respecto al *output*, la figura 7.10 muestra una estimación de la función de densidad de probabilidad conjunta de ambas variables mediante el método *KDE*. Esta figura además expone, que el coeficiente de correlación de *Pearson* es de 0.56, lo que sugiere que efectivamente existe una relación lineal positiva.

No se ha incluido en la figura 7.9 a *Followers*. Para su análisis se deberá aplicar un filtro previo más al dataset para asegurarnos de que el número indicado de seguidores no difiere demasiado de su valor real en el mismo año que tuvieron lugar los ingresos. Por ello se ha establecido el umbral en el año 2014, quedando un total de 14569 instancias. Para analizar la correlación de las variables de esta variación del dataset se utilizará un mapa de calor (figura 7.11). Estas son algunas de las conclusiones:

- *Employees* vuelve a ser el atributo más relevante, llegando a marcar un coeficiente de correlación de *Pearson* de 0,6.
- Existe también cierta correlación entre los atributos *Followers* y *Employees*. Aunque lo óptimo en general sea tener atributos independientes, que este no sea el caso no tiene por qué suponer un problema, y en definitiva repercutirá de maneras distintas según el algoritmo que se utilice.
- En todos los casos las correlaciones son positivas salvo entre *Followers* y *Age*, que no existe ninguna relación.
- Es de especial interés conocer el tipo de relación que existe entre el *output Revenue* y *Followers*, atributo que a priori parecía que iba ser muy útil. En el mapa de calor se muestra que

efectivamente hay una correlación no despreciable. Resulta interesante analizar visualmente este hecho, para lo cuál ayuda la figura 7.12. Dicha imagen cuenta con una recta que estima un modelo de regresión sobre los datos y, si bien indica que con un coeficiente de correlación de *Pearson* de 0,2 la relación no es muy fuerte, demuestra que se trata de un atributo que puede ser útil para la regresión.

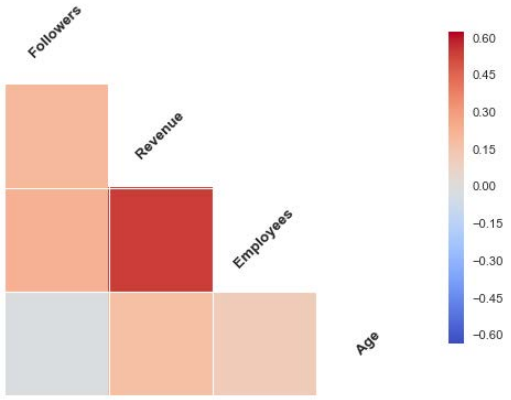


Figura 7.11: *Heat map* de *Employees*, *Age*, *Followers* y *Revenue*

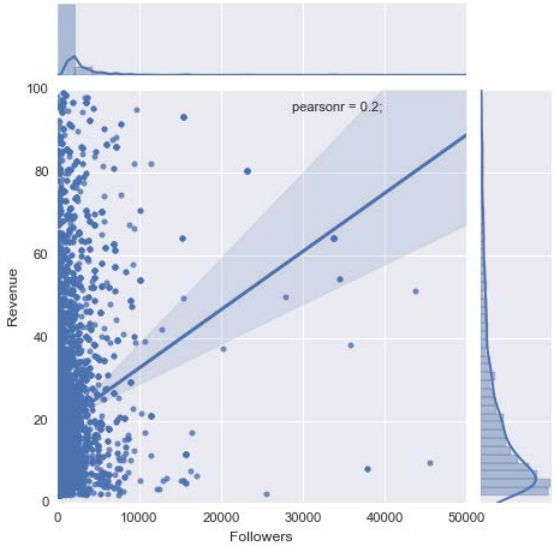


Figura 7.12: *Joint plot* con modelo de regresión lineal de *Followers* respecto a *Revenue*

7.1.5. Modelado y evaluación

Se probarán varios modelos de regresión distintos (principalmente los explicados en la sección 3.3). Como todos los modelos a probar ya existen y están perfectamente implementados en *scikit-learn* la fase de modelado se reduce a decidir qué tipo de modelos probar. Dado que se dispone de importantes recursos computacionales, se probarán varios, desde máquinas de soporte vectorial con *kernel* lineal hasta meta-estimadores.

Estimando el modelo

Dado que el atributo *Followers* ha mostrado tener cierta relevancia, será incluido en el modelo. Por tanto, en la primera prueba ha realizar se utilizará el dataset filtrado a partir de 2014 (14569 compañías). El test a realizar consistirá en entrenar un conjunto de algoritmos distintos para posteriormente evaluar los resultados y seleccionar uno.

Para evaluar los resultados de las pruebas se medirán predicciones a validación cruzada. Para que las pruebas reflejen la capacidad generalizadora del modelo real suele bastar con usar 3 *folds*, pero como el dataset no es excesivamente grande y se tienen bastantes recursos computacionales, se ejecutarán 5.

En la siguiente tabla se presentan los resultados de la prueba:

Primer test de estimadores de los ingresos anuales					
Regresor	Parámetros	Tiempo (s)	Error medio absoluto	R^2	Varianza explicada
Lasso	α=1 selection=cyclic	15.35	16.33	-7.16 %	-7 %
ElasticNet	α=1 l1_ratio=0.5	15.86	16.21	-6.55 %	-6.47 %
SVR	ϵ=0.1 C=1 kernel=linear	870.5	12.55	-9.9 %	13 %
SVR	ϵ=0.1 C=1 kernel=rbf	823.7	14.5	-19.78 %	-5.24 %
Ridge	α=1	16.64	12.3	30.5 %	31.32 %
DecisionTree Regressor	criterion=mse max_features=n_features	30.1	8.3	62.68 %	62.7 %
DecisionTree Regressor	criterion=mse max_features=log2	14.31	11.7	40.35 %	40.57 %
DecisionTree Regressor	criterion=mse max_features=sqrt	14.66	8.9	52.49 %	52.6 %

Se han sacado las siguientes conclusiones de esta evaluación:

- Se observa en primer lugar que el único modelo **lineal** que no ha tenido unos resultados malos es la regresión *Ridge*. Aparte de tener un error absoluto medio ligeramente más bajo, tanto el coeficiente de determinación R^2 como la cantidad de varianza explicada son bastante más altos.

- Por su parte, los árboles de regresión han conseguido las mejores marcas en todas las métricas. En concreto, permitiendo que el modelo considere todos los atributos existentes a la hora de ramificarse. El riesgo de esta configuración era acabar teniendo un modelo con una alta varianza (se recuerda que es la principal desventaja de estos algoritmos), pero los resultados expuestos tras hacer la evaluación cruzada demuestran que se ha mostrado como un modelo robusto ante el sobreajuste.

- De las dos conclusiones anteriores se induce que la relación que existe entre los ingresos anuales y todos los atributos utilizados se estima bastante mejor con modelos no lineales que con modelos lineales.

- Llama la atención también cómo las máquinas de soporte vectorial han tardado aproximadamente 50 veces más de tiempo en ser entrenadas que el resto de modelos, teniendo en cuenta que el dataset utilizado no es especialmente grande (en el contexto del *machine learning*).

- Respecto a las máquinas de soporte vectorial, existe toda una ingeniería de pruebas posibles en relación a la elección del *kernel trick* en conjunción con el parámetro regularizador. En regresión, uno de los procesos de pruebas estandarizados es probar primero con un *kernel* lineal y si no da buenos resultados, probar con el *kernel* de función de base radial. Esta prueba es la que se ha efectuado y el rendimiento no ha sido el adecuado. El diagnóstico es de un sobreajuste del criterio de selección del modelo, y es que las máquinas de soporte vectorial que hacen uso del *kernel trick* son muy sensibles a ello [CT10].

- Se comprueba también que en la mayoría de los casos la varianza sesgada es similar a la no sesgada, porque la métrica R^2 y *Varianza explicada* son bastante similares.

El modelo a seleccionar es, entonces, un árbol de decisión. Se sabe que una característica muy positiva de este estimador que lo diferencia del resto es que una vez entrenado, puede hacer explícito todos los razonamientos aprendidos.

Afortunadamente, la implementación del *CART* de *Scikit-learn* utilizada cuenta con la funcionalidad de visualizar el árbol. Ofrece una sencilla interfaz haciendo transparente al usuario la utilización que realiza internamente de las librerías gráficas *Pydot* y *Graphviz*. En la figura 7.13 se muestra dicho árbol. Como el formato de esta memoria no permite su visualización completa, se expone un fragmento del mismo. El nodo raíz del árbol es el número de empleados. Además, este atributo es usado en varias ramas del árbol debido a su importancia y eficacia a la hora de separar los datos. Se ha etiquetado cada *tag* para aclarar de qué tipo se trata (*Tag*: actividad de la empresa, *GTag*: localización y *Domain*: dominio de la página web). Aparte de *Employees*, *CART* opta a menudo por utilizar *Followers* y *Age* a la hora de ramificarse. Por otro lado, las etiquetas de dominio son menos frecuentes en el árbol.

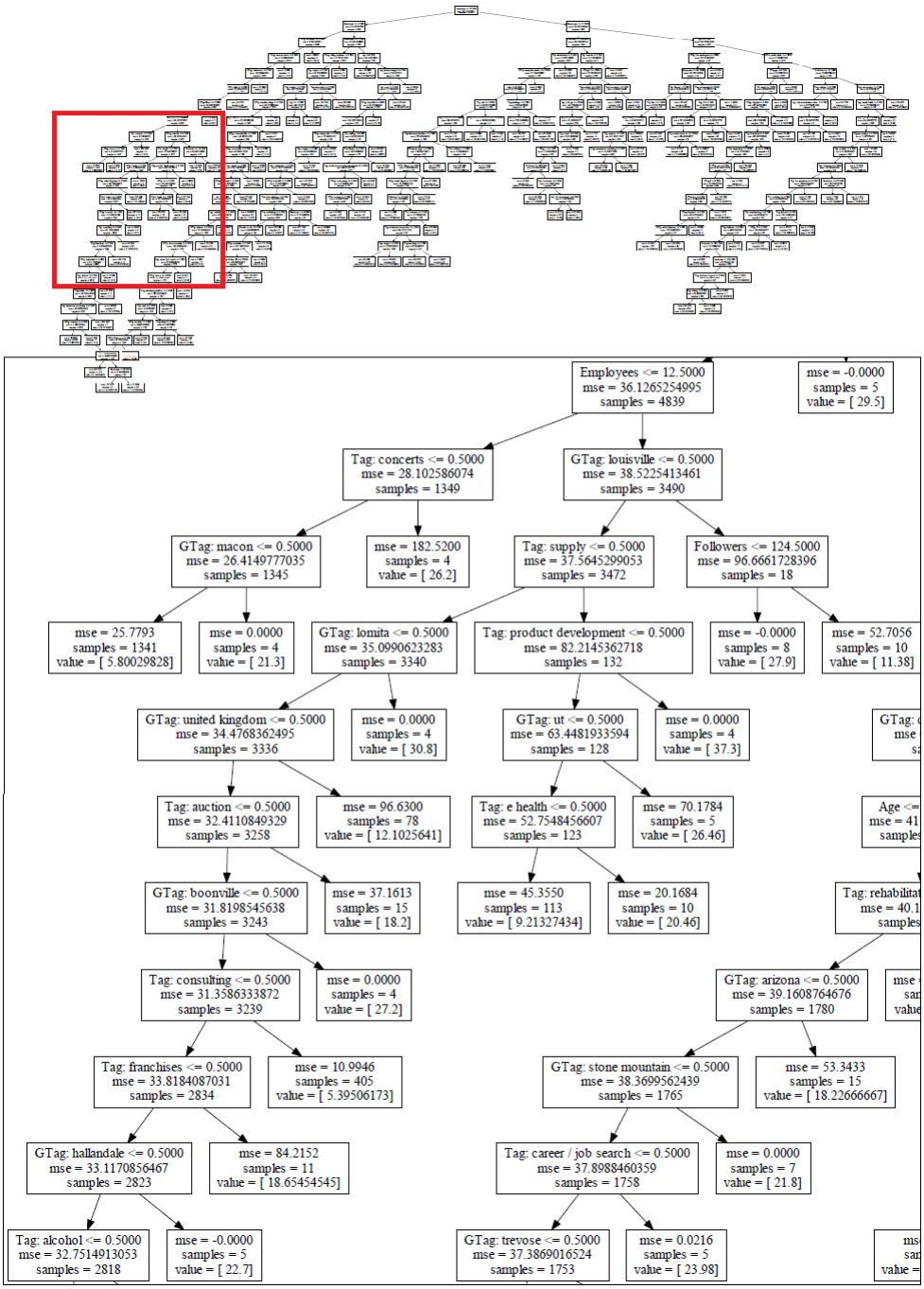


Figura 7.13: Fragmento del *CART* entrenado

Elección del modelo

Ahora que, tras esta evaluación se sabe que el tipo de estimador más adecuado es un árbol de decisión, se pondrán a prueba algunos meta-algoritmos (normalmente basados en la utilización de conjuntos de árboles, como se explicó en la sección 3.3.2) con el objetivo de conseguir mejores resultados. La tipología de las prueba será idéntica al caso anterior, sólo que en este caso los algoritmos a probar serán distintos.

Debido a que los meta-estimadores requieren bastantes más recursos (sobretodo computacionales) que por ejemplo un árbol de decisión individual, se utilizará el parámetro *n_jobs* del método *estimator.fit(X,Y)* para distribuir el trabajo en los procesadores de la máquina. Si se le da un valor de -1, se utilizarán todas las CPUs. Es importante recalcar esto porque un parámetro específico de los meta-estimadores en *Scikit-learn* es *n_estimators*, que indica el número de estimadores que compondrán el meta-estimador, afectando directamente a los recursos que se vayan a consumir durante el proceso. Dado que se dispone de una potente máquina, no se escatimará y se construirán muchos estimadores.

Estos son los resultados de esta prueba:

Segundo test de estimadores de los ingresos anuales					
Regresor	Parámetros	Tiempo (s)	Error medio absoluto	R^2	Varianza explicada
AdaBoost	n_estimators=700 learning_rate=1	103	17.45	4.33 %	28.78 %
AdaBoost	n_estimators=500 learning_rate=0.5	38	16.28	12.8 %	30 %
GradientBoosting	n_estimators=500 criterion=mse loss=ls	926	9.74	54.57 %	54.7 %
GradientBoosting	n_estimators=700 criterion=mse loss=lad	1251	9.63	38.6 %	41.53 %
ExtraTrees	n_estimators=700 criterion=mse bootstrap=False max_features=n_features	15572	5.2	65.7 %	65.76 %
ExtraTrees	n_estimators=500 criterion=mse bootstrap=True max_features=n_features	8052	6.1	63 %	63.34 %
RandomForest	n_estimators=500 criterion=mse bootstrap=False max_features=n_features	6577	8.1	56.77 %	57.5 %
RandomForest	n_estimators=700 criterion=mse bootstrap=True max_features=n_features	10457	6.23	63.3 %	63.39 %
Bagging	n_estimators=700 bootstrap=False	5950	7.9	44.58 %	50.1 %
Bagging	n_estimators=500 bootstrap=True	5223	7.62	45.4 %	51.7 %

Se han sacado las siguientes conclusiones de esta evaluación:

- Lo primero que se observa es que los resultados no sólo son mejores a cualquier regresor lineal de los probados anteriormente, sino que también hay una mejora significativa con respecto al árbol de decisión. Es decir, existe suficiente evidencia para afirmar que en este problema se cumple la premisa de los meta-estimadores: el conjunto de estimadores débiles/aleatorios ha vencido al estimador 'experto'.
- En este problema, parece que los meta-estimadores de la familia de 'comité' (voto promedio) son mejores predictores que los que utilizan técnicas de *Boosting*.
- El meta-estimador de Árboles extra-aleatorios es el que más capacidad predictiva demuestra. La reducción de la varianza mediante la aleatorización extrema de este algoritmo ha sido útil.

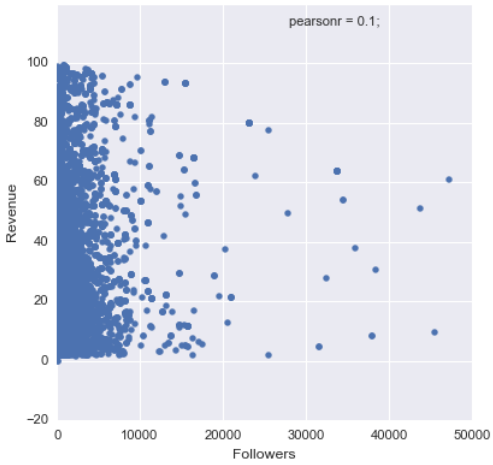


Figura 7.14: Correlación entre *Followers* con ruido y *Revenue*

- No se tiene evidencia de cómo afecta el muestreo aleatorio uniforme con reemplazo (*bootstrap*) al rendimiento del predictor.
- Se puede afirmar casi de manera unánime que aumentar el número de estimadores (árboles) en cada modelo repercute positivamente en la capacidad predictora y como era de esperar, negativamente en los recursos utilizados. Destaca especialmente precisamente el modelo que mejor resultados ha obtenido: *ExtraTrees* sin *bootstrap* y con 700 estimadores, que tardó en ejecutarse 42 horas.

Antes de finalizar la selección del modelo final que se implementará en el sistema, es conveniente considerar que en las pruebas efectuadas hasta ahora se estaban utilizando aproximadamente el 40 % de todos los datos recopilados para que tuviese sentido poder utilizar la variable *Followers* en el modelo. Por ejemplo, no resulta a priori correcto que para algunas de las empresas con ingresos del año 2015 se muestren los seguidores que tenían en ese mismo año mientras que para otras se muestre el de 2010. ¿O sí? Una cosa es segura: la última palabra la tiene el modelo. Da totalmente igual el nombre que se le ponga a la columna, el estimador siempre la considerará tanto en cuanto la considere útil. Es decir, si se utiliza todo el dataset, mostrando para algunas empresas los seguidores que tuvieron en el mismo año que los ingresos indicados y para otras en periodos distintos, la variable *Followers* será considerada por el modelo tanto en cuanto le sea útil según sus criterios (correlación con el *output*, ganancia de información, separación de instancias etc.). Por ello, antes de tomar la decisión final sería conveniente analizar en qué medida incluirla va a ser beneficioso o perjudicial para las estimaciones.

En primer lugar, si analizamos visualmente la correlación que existe entre esta variable y el *output* en todo el dataset (ver figura 7.14), como era de esperar, están muy poco correladas. No obstante, los meta-estimadores son capaces de utilizar las relaciones no lineales entre las variables y esto no queda reflejado en la figura 7.14. Introducir la variable *Followers* olvidándose de la restricción temporal introduciría ruido en el modelo pero es posible que a cambio se aportase información útil. Para tener algo de ayuda en esta decisión sería conveniente echar un vistazo a cómo es la curva de

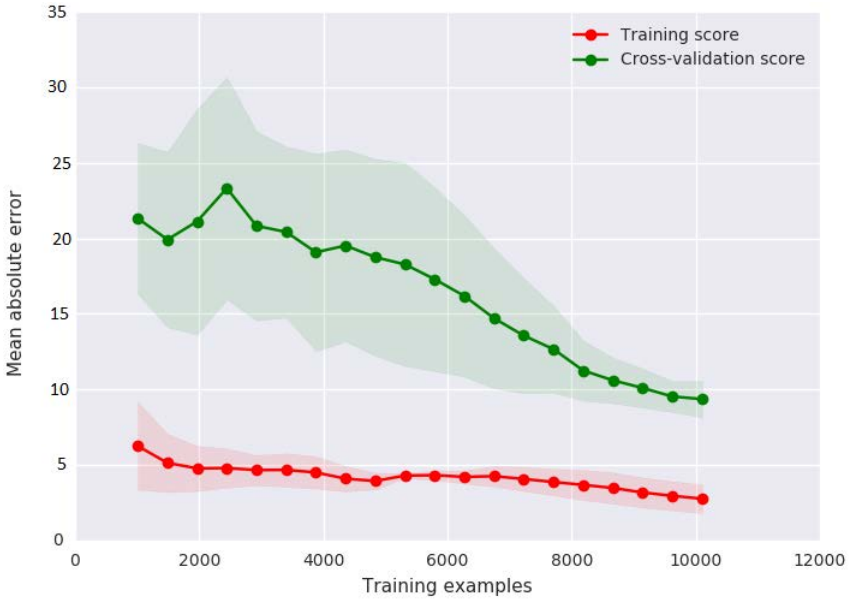


Figura 7.15: Curva de aprendizaje del regresor *ExtraTrees* sin *bootstrap* y con 700 estimadores

aprendizaje del modelo hasta ahora construido. En la figura 7.15 se ha dibujado como evoluciona el error de test y el error de entrenamiento según se añaden más ejemplos al dataset (no salen los aproximadamente 15000 ejemplos debido a que el 30 % se quedan fuera para medir el error). El modelo entrenado es el que mejor resultados dio en la prueba anterior: *ExtraTrees* sin *bootstrap* y con 700 estimadores. En la gráfica dibujada, las medidas de error han sido calculadas mediante validación cruzada de 4 *folds*. El sombreado que tiene alrededor cada curva indica lo dispares que han sido las distintas mediciones en cada evaluación. Como es normal, a medida que aumentan los ejemplos las variaciones disminuyen y las distintas mediciones en cada punto para cada curva tienden a converger (hay menos discrepancia). A continuación se exponen las conclusiones más relevantes que se pueden extraer del análisis de esta figura:

- Lo más relevante de la figura 7.15 es sin duda la claridad con la que se aprecia cómo va disminuyendo el error de validación cruzada o test a medida que se aumentan el número de empresas con el que se entrena el modelo. Esto sugiere que seguir aumentándolo es una buena idea.

- No hay indicios de sobreaprendizaje. Como se ha comentado, los meta-estimadores son buenas soluciones a la hora de reducir la varianza, y *ExtraTrees* es de los algoritmos más extremistas en este aspecto.

- Posible sesgo del modelo. Según se aumenta el número de instancias el error de entrenamiento parece no disminuir o disminuye muy poco. Si efectivamente este error no variase por muchas instancias añadidas al entrenamiento, se concluiría que el modelo tiene sesgo. La manera

de solucionar este hipotético sesgo sería enriquecer el modelo con más atributos que sean relevantes.

Ante esta situación, la decisión que se toma es la de ampliar el dataset independientemente de que la columna de seguidores pierda su significado. De hecho, más que perderlo, se puede decir que cambia:

> **Followers**: Número de seguidores en la red social *Linkedin* que la empresa tuvo en el mismo año en el que se computaron sus ingresos.

> **Followers**: Número de seguidores en la red social *Linkedin* que la empresa tuvo en algún momento.

La nueva definición es más vaga pero es consistente con la manera de almacenar los datos. A continuación se procede a efectuar la prueba con todo el dataset. Como ya se conoce cuáles son los modelos que funcionan mejor, y la carga computacional es mayor al entrenar con más del doble de ejemplos, sólo se probarán tres configuraciones de meta-algoritmos (los tres perteneciente a la rama de métodos de 'comité'). Estos son los resultados de esta costosa prueba que duró 4 días:

Tercer test de estimadores de los ingresos anuales					
Regresor	Parámetros	Tiempo (s)	Error medio absoluto	R^2	Varianza explicada
ExtraTrees	n_estimators=700 criterion=mse bootstrap=False max_features=n_features	166724	3.93	79.75 %	79.74 %
RandomForest	n_estimators=700 criterion=mse bootstrap=True max_features=n_features	118230	5.9	74.91 %	74.87 %
Bagging	n_estimators=700 bootstrap=False	75631	5.76	74.31 %	74.17 %

- Existe una mejoría significativa en todos los casos.
- Una vez más, el mejor meta-algoritmo ha sido el *ExtraTrees* sin *bootstrap* y con 700 estimadores, llegando a alcanzar una métricas aceptables.

Finalmente y por alcanzar los mejores resultados de entre todos los probados, se elige al meta-algoritmo *ExtraTrees* sin *bootstrap* y con 700 estimadores como modelo para estimar los ingresos anuales y pasará entonces a ser una pieza fundamental en el método de valoración.

Ya se ha cumplido el principal objetivo de este apartado. No obstante, antes de cerrarlo sería muy positivo poder visualizar el conocimiento que el modelo ha aprendido. Lamentablemente, los meta-estimadores son una 'caja negra' comparados con los árboles de decisión individuales. Por otro

lado, algo que sí se puede hacer es listar la importancia que éste ha dado a cada uno de los atributos. Una vez más, la implementación de *Scikit-learn* cuenta con esta funcionalidad, de tal manera que dicho listado se encuentra almacenado como un atributo del objeto que representa al modelo una vez este ha sido entrenado. Siguiendo uno de los muchos ejemplos de uso en el manual de dicha librería [14a], se ha construído una visualización de dicho listado (figura 7.16), incluyendo el top 50 atributos útiles. Conviene recordar que el modelo cuenta con 700 estimadores (árboles) y debido a la aleatorización extrema, cada uno de ellos ha podido considerar con más o menos importancia cada atributo. Por ejemplo, algunos han podido utilizar más frecuentemente para separar los datos que el dominio de la página web sea *.com* o no y otros que la empresa esté situada en EEUU o no. El nivel de consenso alcanzado entre los distintos árboles del modelo en lo que importancia otorgada a los atributos se refiere se conoce como variabilidad inter-árbol y se representa en la figura 7.16 como una **desviación típica** (líneas) de los **valores medios** indicados (barras). De esta visualización se sacan las siguientes conclusiones:

- El atributo *Employees* se vuelve a confirmar como el más importante de la regresión.
- Sorprendentemente, a pesar del ruido introducido en el atributo *Followers*, éste es el tercero más importante, por lo que la decisión de mantenerlo fue positiva.
- Hay bastante indecisión entre los 700 estimadores respecto a la importancia a otorgar al hecho de que una empresa tenga la geo-etiqueta 'united states', pero en promedio se sitúa como la cuarta característica más importante.
- Por un lado, las etiquetas de tipo de actividad de la empresa (T) y las de localización se utilizan por igual, mientras que la del dominio de la *url* menos, tan sólo apareciendo en el ranking *.com* (quinta posición) e *it* (vigésimo primera posición).

Extrayendo conocimiento de los datos

Aprovechando el modelo generado se procede a continuación a realizar un análisis más de los datos desde un punto de vista práctico, para ayudar al cumplimiento del objetivo 5 de la sección 1.3. Hay que tener en cuenta que en la figura 7.16 se muestran importancias, pero en el caso de los árboles, importancia no implica correlación positiva o negativa directamente, por lo que ahora que se conocen cuáles son los atributos más importantes de la regresión, sería interesante dibujar un mapa de calor con ellos (figura 7.17). Esto no se hizo antes porque a priori no se sabía cuáles, de las 1470 etiquetas de actividad de empresa, 4655 de localización y 97 dominios eran atributos importantes. De esta nueva figura se puede decir que:

- Ningún atributo muestra una correlación notablemente negativa con el *output Revenue*.
- Después de *Employees*, se tiene que existe una correlación positiva entre que una empresa sea de Reino Unido o se dedique al software y sus ingresos anuales.
- No obstante, no se cumple que los atributos más importantes (figura 7.16) sean los más (negativa o positivamente) correlados.
- Una correlación negativa entre dos atributos categóricos indica que tienden a ir totalmente separados el uno del otro. Hay algunos casos que ejemplifican esto perfectamente:
  - Empresas de servicios y empresas manufactureras.

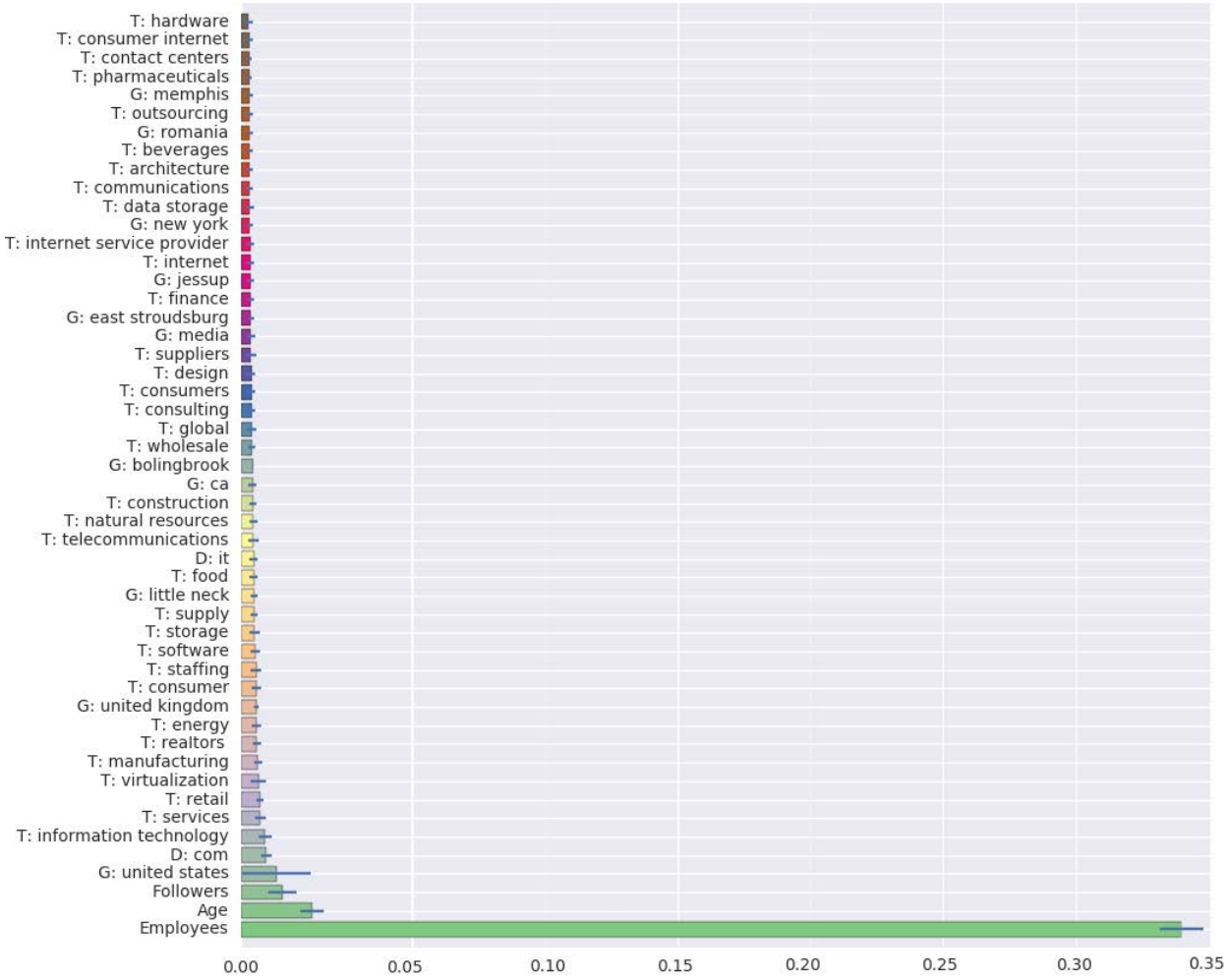


Figura 7.16: Histograma con la importancia otorgada a los atributos del modelo por los 700 estimadores

  - Empresas de servicios y empresas. minoristas
  - Empresas manufactureras y empresas minoristas.

- De manera equivalente, hay correlaciones positivas entre etiquetas que también tienen sentido, y otras que son interesantes:
  - Las empresas del sector de Tecnologías de la Información están muy relacionas con software y en menor medida con los servicios.
  - Las empresas de servicios están relacionadas con el consumidor.
  - Las empresas de provisionamiento y almacenamiento.
  - Las empresas con presencia en EEUU también la suelen tener en Reino Unido.
- Las empresas que tienen mayor presencia en la red social *Linkedin* tienden a ser las relacionadas con la virtualización, seguidas de lejos por las de energía y almacenamiento. También hay una notable correlación positiva de este atributo Y las empresas estadounidenses.

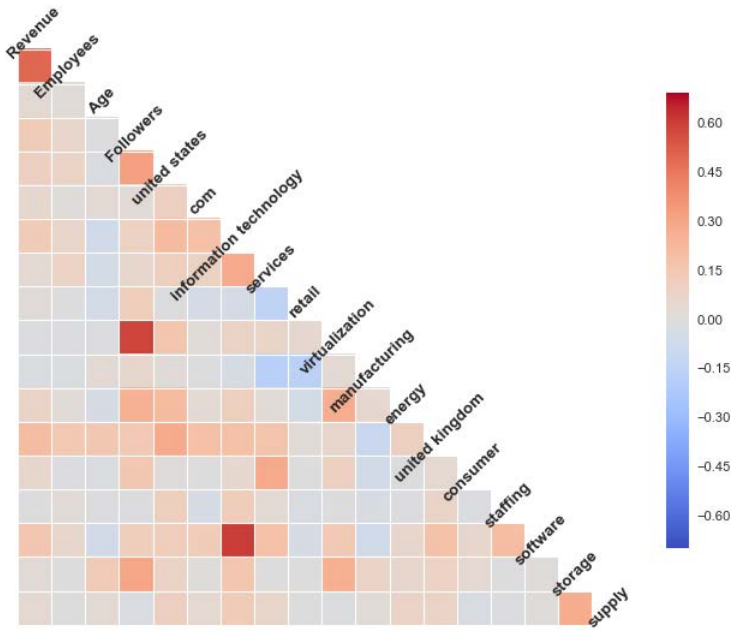


Figura 7.17: Mapa de calor de los atributos más importantes en el modelo final

7.2. Estimando los flujos de caja operativos

7.2.1. Entendimiento del problema y objetivos

El problema a resolver ahora es conseguir de manera automática los flujos de caja operativos (o OCF, de *operative cash flows*) a partir de sus ingresos anuales estimados en la sección anterior. La diferencia entre los ingresos anuales y estos flujos se reduce a los gastos operativos. Es muy difícil obtener acceso a estos gastos para cada empresa, especialmente cuando esta es muy joven.

No obstante, es posible y sería razonable que aquellas empresas con el mismo tipo de actividad tuviesen los mismos gastos operativos respecto a sus ingresos.

$$\frac{OCF}{Revenue} \tag{7.1}$$

Si efectivamente existiese una correlación entre este ratio y la actividad de una empresa (sector, industria etc.) una buena estrategia sería crear un modelo que aprendiese cuáles son los ratios para cada tipo de empresa.

Dado que ni en el blog de Damodaran ni en ninguna otra fuente se han encontrado medias de este ratio por sector e industria que pudiesen arrojar luz al asunto, será preciso calcularlas. El primer objetivo por tanto será obtener un suficientemente grande conjunto de datos de empresas de donde poder extraer los ratios. Como no de todas las empresas se puede conocer con la misma facilidad este ratio, será necesario acudir a fuentes abiertas al público y de fácil acceso.

7.2.2. Obtención y entendimiento de los datos

Obtención de los datos

Habrá que implementar un *web scrapper* específico para obtener los datos de *yahoo finance*. La tecnología utilizada será la misma a la de los *web scrappers* de las secciones 7.1.2 y 5.2.3 por lo que se documentará el trabajo realizado sin ahondar en los detalles de la implementación:

1. En *yahoo finance*, cada compañía está identificada por un código único.
2. El sector de una compañía y su número de empleados están en una vista (general) mientras que los ingresos y flujos de caja operativos en otra (detalles). Esto implica que por cada compañía se van a tener que realizar un mínimo de dos peticiones. Afortunadamente, cada tipo de vista está identificada en la `url`, la cuál recibe cómo parámetro el código de la compañía:
   a) General: `finance.yahoo.com/q/pr?s=<COMPANY_CODE>`
   b) Detalles: `finance.yahoo.com/q/ks?s=<COMPANY_CODE>`
3. El problema siguiente es cómo obtener un listado de los códigos de empresas a partir de los cuáles empezar a lanzar las peticiones a cada tipo de vista. Una de las prioridades es contar con un número suficientemente grande de compañías distribuido por todo el abanico de sectores. Por tanto, se utilizará el explorador de industrias `biz.yahoo.com/ic/ind_index.html` de *yahoo finance* donde yace una lista de las 208 industrias clasificadas en 9 sectores, existiendo en cada elemento un enlace a una vista donde se listan varios códigos de empresas pertenecientes a dicha industria.
4. Con la estrategia planteada, se definen los 3 módulos (*scripts*) del *web scrapper*:

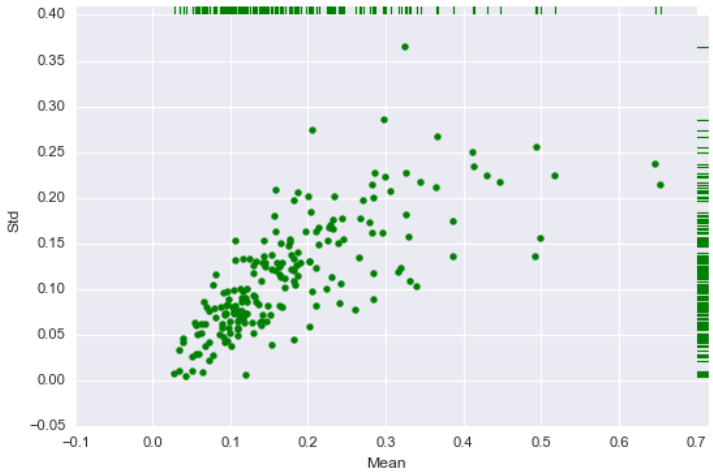


Figura 7.18: Media y desviación típica de los OCF/Revenue de cada sector de *Yahoo Finance*

4.1. Escanear al completo la clasificación de industrias mientras se van anotando los códigos de empresas.

4.2. Descargar todos los `html` correspondientes a la vista general.

4.3. Descargar todos los `html` correspondientes a la vista de detalles. Tanto en este como en el anterior paso se lanzan las peticiones con esperas de 20-40 segundos intermedias.

4.4. Escanear todos ficheros `html` descargados y construir una tabla (objeto *DataFrame*) a partir de los mismos.

Entendiendo los datos

Una vez obtenidos los datos se puede proceder a su análisis. Un factor clave que hay que comprobar es cómo de distintos son los ratios *OCF/Revenue* entre las distintas industrias, ya que dependiendo de lo estable que sean los ratios se tomará una decisión u otra para alcanzar los objetivos. Para llevar a cabo este análisis es preciso:

1. Limpiar nulos, ya que no todas las compañías contaban con estos indicadores financieros (sólo 10000 de ellas tienen ambos valores).

2. Calcular el ratio para cada empresa. Con la función *apply* de *Pandas* generamos la nueva columna dividiendo los valores de las otras dos.

3. Utilizar la función *groupby* de *Pandas* para agrupar las empresas por industria y sector y calcular para cada subset la desviación típica y la media de la columna *ratio*. Se ha utilizado posteriormente *Seaborn* para visualizar las medias y desviaciones típicas de los distintos grupos en un diagrama de puntos. Dicho diagrama se puede encontrar en la figura 7.18.

4. Utilizar otra vez la librería *Seaborn* para hacer una representación más informativa de la estabilidad del ratio de cada sector (pero no de todos ellos) mediante diagramas de cajas con bigotes (ver figura 7.19).

De ambas figuras se pueden sacar algunas conclusiones:

- De la figura 7.18 se deduce que a más media, más desvicación típica, algo normal ya que se están midiendo ratios.
- Para la mayoría de sectores los flujos de caja operativos no suponen más del 30 % de los ingresos (percentil 75 es 0.29).
- En la muestra aleatoria de sectores de la figura 7.19 se observa que la mayoría tienen la mediana parecida.
- No obstante, en algunos sectores los bigotes son bastante más grandes que los cuartiles intermedios, como es el caso de *Information Technology Services* (sector popular entre *startups*), existiendo tanto empresas que sólo consiguieron mantener como flujos de caja el 0,4 % de sus ingresos anuales y otras prácticamente la totalidad de ellos.

En conclusión, no existe suficiente estabilidad en los ratios de un mismo sector como para limitarse exclusivamente a utilizar sus medias en el modelo. Por tanto, la opción que queda es intentar ajustar los ratios mediante un regresor. Para crear el regresor se tienen que seleccionar los atributos primero. No hay muchos donde elegir, ya que no se debe olvidar que sólo se pueden utilizar campos que se sepa con certeza que se va a disponer de ellos en el futuro. Se escogen los siguientes:

- **Tags**: Se corresponde con las etiquetas extraídas con el extractor de árbol con reglas a partir de la descripción de la empresa (aparece en la vista general de cada compañía de *Yahoo Finance*). Para maximizar la relevancia de este atributo, la extracción de etiquetas se realizará a partir de un campo nuevo generado a partir de la concatenación de la descripción, industria y sector de cada empresa.
- **Gtags**: Correspondiente con el país y ciudad de la empresa (también existente en la vista general)
- **Revenue/Employees**: Con el afán de incluir el máximo número de atributos con potencial relevancia se añade como atributo los ingresos anuales por empleado de la compañía. El motivo por el que no se incluyen directamente los empleados (que se sabrán con certeza) o los ingresos (que se estimarán previamente con el otro regresor) es que las empresas del dataset de *Yahoo Finance* son públicas y muchísimo más grandes que la gran mayoría sobre la cuál se requiere realizar valoraciones. Si se entrenase y evaluase el modelo con instancias cuyos atributos son de varios órdenes de magnitud superiores a los de las instancias objetivo muy probablemente se encontrarían resultados con poco sentido difícilmente diagnosticables.

7.2.3. Preparación de los datos y modelado

Transformando los inputs

Estos son los preprocesados y transformaciones que será necesario aplicar a los datos de entrada antes de entrenar el modelo:

- Esta vez, se hará el mismo tratamiento de los *Tags* y los *GTags*: aplicación de la transformación *TF IDF* con un vocabulario predeterminado por las mismas etiquetas, aprovechando que ya se sabe cuáles son las palabras clave que definen la actividad de la empresa, de tal manera

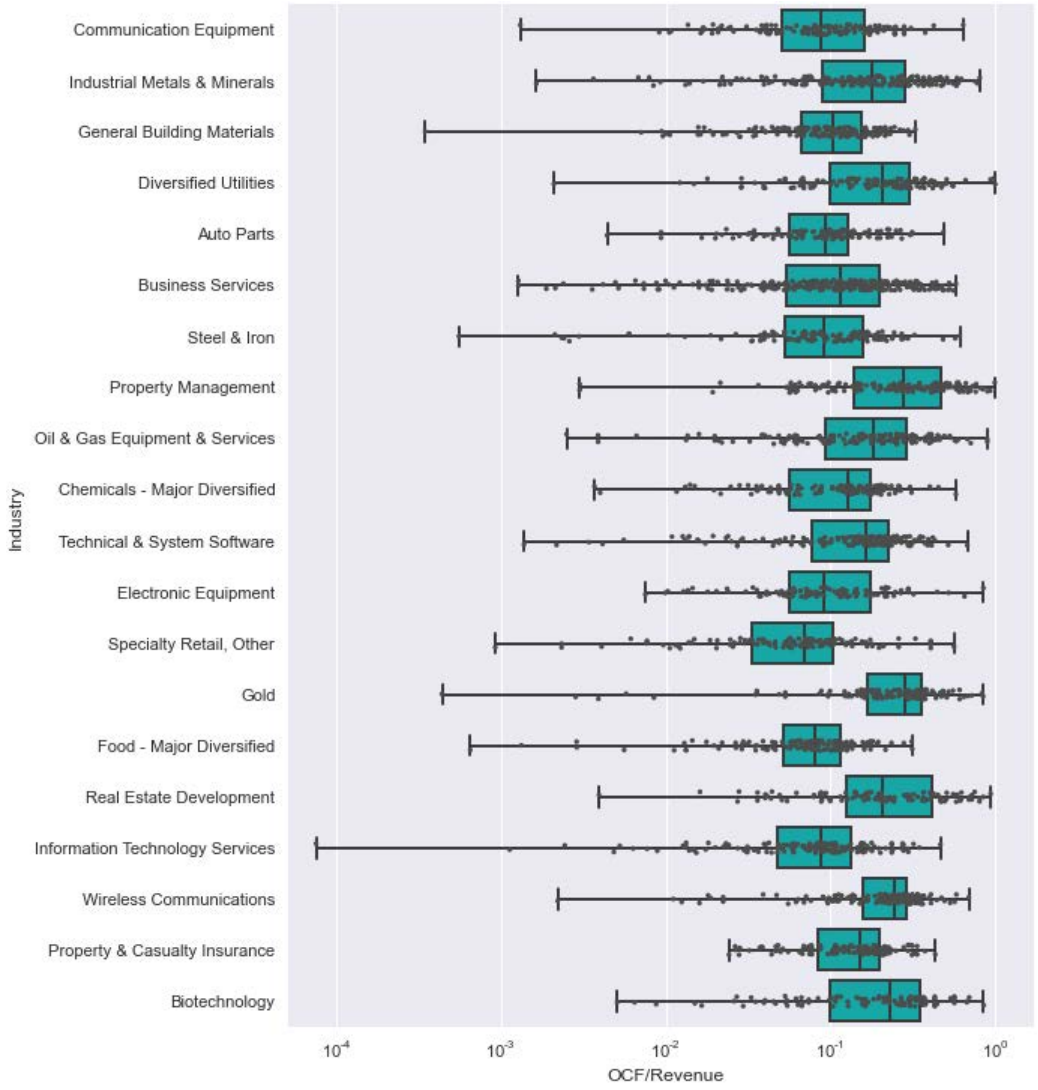


Figura 7.19: Diagrama de caja y bigotes del OCF/Revenue de algunos sectores de *Yahoo Finance*

que por ejemplo sólo contemplará los n-gramas que le sean indicados explícitamente. Así el modelo será más eficiente en términos generales. Además, se aplicará el escalado sublineal al término **TF**.

- Se normalizará el valor del ratio *Revenue/Employees*. De esta manera se asegura que todos los valores numéricos transformados del dataset vayan a estar entre 0 y 1. Si fuese de otra manera podrían ocurrir problemas en algunos modelos dependiendo del algoritmo de aprendizaje.

Sin incluir el ratio **Revenue/Employees** se tiene un dataset de 9831 instancias, pero si se incluye se queda en 6675. Como a priori no se sabe qué es mejor, si utilizar más instancias o incluir este atributo, se generarán ambos datasets.

Elección del modelo

Se trata de un modelo de regresión por lo que se puede reutilizar la selección de modelos de aprendizaje candidatos que se hizo para el regresor de los ingresos. Aunque se sabe de la potencia de meta-estimadores como Selvas Aleatorias o *Bagging*, no se descartará probar modelos más simples como las máquinas de soporte vectorial.

7.2.4. Evaluación

Al estar intentando estimar un ratio, la métrica con la que se evaluarán los modelos será R^2. La tabla 7.1 muestra la comparativa no sólo de los distintos algoritmos sino de los dos datasets que surgen de la ausencia (1) o presencia (2) del atributo **Revenue/Employees**. En todos los casos las métricas se han tomado a partir de una validación cruzada de tamaño 3. De dicha tabla se desprenden las siguientes conclusiones:

- Resulta evidente que en este problema los meta-estimadores ofrecen (menos *AdaBoost*) los mejores resultados. En concreto, y no por mucha diferencia, la puntuación más alta ha sido conseguida por el *ExtraTrees* con muestreo con reemplazo activado, alcanzando un R^2 de 0.567.

- Se concluye que a pesar de entrenar con menos instancias, la inclusión del atributo **Revenue/Employees** ha sido positiva, por lo que se incluirá en el modelo final.

- Por otro lado, las puntuaciones de los algoritmos más simples han sido bastante bajas, dando en algunos casos valores de R^2 negativos (lo que indica que la capacidad generalizadora es peor que estimar siempre la media). El único que llega a explicar parte la variación de la salida es la máquina de soporte vectorial *SVR* con *kernel* lineal marcando un 0.35.

Los resultados no son magníficos. No obstante, dado que se consigue explicar más de la mitad de la varianza de la proporción de los flujos de caja operativos sobre los ingresos, se decide que se integrará el regresor en el modelo. Lo que se buscaba en esta sección es conseguir ajustar los ingresos para que se parezcan más a los flujos de caja con los que poder valorar, y con el modelo generado se consigue cierta capacidad de generalización, mucho mejor que simplemente tomar el ratio medio de cada sector.

Evaluación de regresores del ratio OCF/Revenue			
Regresor	Parámetros	R^2 (1)	R^2 (2)
Lasso	α=1 selection=cyclic	-1.2	-0.022
ElasticNet	α=1 l1_ratio=0.5	0.08	-0.04
SVR	ϵ=0.1 C=1 kernel=linear	0.24	0.35
SVR	ϵ=0.1 C=1 kernel=rbf	0.08	0.09
Ridge	α=1	0.22	0.10
AdaBoost	n_estimators=500 learning_rate=1 loss=linear	-0.027	0.017
AdaBoost	n_estimators=500 learning_rate=0.5 loss=linear	-0.5	-0.03
ExtraTrees	n_estimators=500 criterion=mse bootstrap=False max_features=n_features	0.447	0.556
ExtraTrees	n_estimators=500 criterion=mse bootstrap=True max_features=n_features	0.433	0.567
RandomForest	n_estimators=500 criterion=mse bootstrap=False max_features=n_features	0.46	0.551
RandomForest	n_estimators=500 criterion=mse bootstrap=True max_features=n_features	0.46	0.558
Bagging	n_estimators=500 bootstrap=False	0.459	0.542
Bagging	n_estimators=500 bootstrap=True	0.407	0.559

Cuadro 7.1: Evaluación de regresores del ratio OCF/Revenue

7.3. Deduciendo el CAPEX

Como se ha explicado, para que el método de valoración sea lo más preciso posible, es conveniente descontar flujos de caja libres, para lo cuál hay que descontar el CAPEX a los OCF que resultan de la sección anterior. El razonamiento a seguir en este caso es similar al de la sección 7.2 en cuanto a por qué es mejor hacer una estimación del ratio sobre los ingresos. No obstante, como se explica a continuación, parece razonable que exista una tendencia a que el ratio entre las inversiones en capital entre los ingresos sea estable dentro de un sector. Si se tomase esta suposición, una solución perfectamente válida sería aplicar los ratios medios sin necesidad de gastar recursos en implementar otro regresor como en la sección 7.2.

Estabilidad del ratio CAPEX/Ventas entre sectores

El tipo de industria en el que una empresa involucra su actividad económica determina en gran medida la naturaleza de sus gastos en capitales. Los efectos del *CAPEX* sobre la valoración de una empresa dependen de en qué tipo de inversiones de activos fijos gasta más dinero la compañía. En general, el CAPEX puede ser necesario por motivos de mantenimiento o de crecimiento y producción. Explorando un poco es fácil encontrar con multitud de fuentes que muestran, entre otros indicadores financieros, que porcentaje de las ventas de una empresa se gastan en inversiones en activos fijos en cada uno de las industrias [HG10][14d][14g][Dam16].

Explorando los informes se concluye que se utilizarán las medias de los ratios, ya que parece razonable suponer que son estables dentro de un mismo sector. Además, como se adelantaba, hacerlo así supondrá un considerable ahorro de recursos (dinero y tiempo) respecto a otras soluciones como la de intentar hacer un regresor, cuyas aportaciones, además de no garantizarse que vayan a ser positivas, difícilmente justificarían el tiempo que habría que invertir en su desarrollo e implementación.

Una vez tomada esta decisión, hay que elegir de qué estudio escoger los ratios medios $\frac{CAPEX}{Revenue}$ para su integración en el modelo. Se elegirá el proporcionado por el profesor *Damodaran*, principalmente porque es el que más actualizado está (Enero de 2016), pero a su vez porque proviene de una fuente académica[Dam16]. Por otro lado, el ratio que proporciona *Damodaran* tiene en cuenta la diferencia entre los gastos de capital y la depreciación (*Net CAPEX*), lo que añade precisión al cálculo.

Como se introdujo en 5.3, Damodaran saca medias habitualmente de 95 sectores. En la figura 7.20 se muestran los valores más altos y los valores más bajos. Como se puede observar, el único sector en el que la inversión en capital es mayor a los ingresos anuales se trata de las energías renovables, lo que tiene sentido debido a los altos costes asociados al desarrollo y construcción de una planta energética.

Como también se introdujo en la sección 5.3, la integración de las medias de *Damodaran* en el proceso de valoración principal del sistema será facilitado mediante un *script* en *Python* que se encargará de hacer el matching entre los sectores de *Damodaran* y las actividades empresariales expresadas en forma de múltiples etiquetas con las que se trabaja en este proyecto.

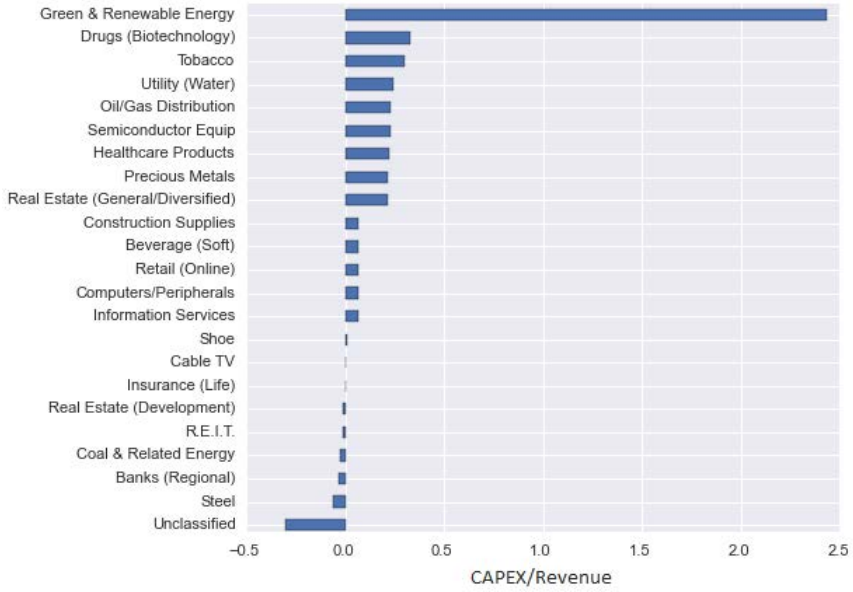


Figura 7.20: Ratio CAPEX/Revenue de algunos sectos según *Damodaran*[Dam16]

Parte IV

Despliegue y Conclusiones

Capítulo 8

Despliegue

Una vez que se ha resuelto cada uno de los subproblemas por separado, es momento de conectar todos los submodelos resultantes en uno único capaz de generar valoraciones de empresas en un sólo paso. Como se explicó en la sección 4.1.2, una de las ventajas del lenguaje *Python* que le hacen distinto del resto de herramientas de *Data Mining* es que resulta eficaz tanto en entornos de experimentación como de producción. Esta característica hace que se ahorre en recursos de implementación, obteniendo un prototipo completo a partir de los modelos experimentales sin tener que invertir demasiado esfuerzo.

Es el objetivo de este capítulo describir en primer lugar la integración del del sistema. Después se pondrá el modelo a prueba con un dataset de empresas objetivo. Se calculará la valoración de las mismas y se llevará a cabo un análisis mediante gráficos para extraer cualquier información que pueda resultar útil.

8.1. Juntando las partes

A continuación se listan los distintos submodelos a modo de recapitulación. También se detallará, según el caso, como se integrarán en el sistema final. A modo de recordatorio se referencia la fórmula de valoración adaptada al problema, donde cada input se corresponde con uno de los subproblemas que se han resuelto:

$$V = \frac{SR \times Revenue \times (Ratio_1 - Ratio_2)}{WACC - g}$$

Determinación de la actividad de la empresa, tags

- **inputs**: (*texto*)

Se resume a una función que toma como entrada un texto (como la descripción de la empresa) y devuelve un conjunto de etiquetas que capturan con cierto nivel de detalle a qué se dedica la empresa. Dada la utilidad de esta función, se han implementado dos versiones de la misma en función del tipo de ejecución para aportar versatilidad:

- **Puntual**: Su versión más sencilla, pensada para extraer etiquetas de manera puntual. Por ejemplo cuando aparece una nueva compañía y sólo se requiere ejecutar la valoración sobre ella. Ejecuta el sistema experto de reglas con normalidad.

- **Lotes**: Pensada para extraer etiquetas de manera masiva. Recibe como entrada una columna de descripciones y entera y paraleliza la extracción entre 30 procesos, mandando a cada uno una porción de la misma. Cada proceso hace a su vez uso de la función *map* de *Pandas* para hacer más eficiente el proceso. Cabe destacar que gracias a esta función se cumple con el requisito 05 del cuadro 5.2. De media tarda en ejecutar 100000 descripciones (de 1000 caracteres cada una aproximadamente) 4 minutos y 9 segundos.

Crecimiento, g

- **inputs**:$(tags)$

Como el conjunto de posibles etiquetas está predefinido (cada vez que se amplíe habrá que entrenar todo otra vez), no habrá que descargarse los datos de tendencias en cada ejecución. Cada mes se descargarán todas las tendencias de todas las etiquetas existentes y se almacenarán en el disco duro del servidor con el indicador de crecimiento ya calculado mediante al fórmula 6.3. De esta manera, cuando se quiera medir el crecimiento de una empresa, bastará con leer los ficheros y puntuaciones correspondientes a las etiquetas de actividad de la compañía en cuestión y calcular el promedio.

WACC

- **inputs**:$(tags)$

En la primera versión se utilizará como WACC las medias de Damodaran. Por tanto, esta función recibirá un conjunto de etiquetas que definan la actividad de la empresa y mediante el proceso de *matching* explicado en la sección 5.3, se devolverá el indicador.

SR

- **inputs**:$(tags, age)$

En esta fase de la ejecución se determinará el *Success Rate* de la empresa. Para ello, se tomarán como entrada las etiquetas de actividad empresarial y la edad. No obstante, es importante mencionar que dado que los nombres de las industrias que se utilizan en 6.3 vienen de distinta fuente (distinta también de las industrias de Damodaran), es necesario introducir un paso intermedio en el que se traduce el conjunto de etiquetas. Para ello se ha tenido que generar un pequeño fichero de 11 líneas (tantas como sectores con distinto riesgo se diferencian) donde se busca maximizar el número de coincidencias acertadas mediante un conjunto reducido de reglas. Para hacer este *matching* se utiliza el sistema extractor de reglas normal pero utilizando un fichero `.csv` de reglas distinto (es un parámetro de la función).

Revenue

- **inputs**: $(tags, gtags, web\_domain, followers, employees, age)$

Merece una mención especial el atributo correspondiente a las etiquetas de localización geográfica, ya que las etiquetas utilizadas en el dataset objetivo para identificar países, estados, provincias, ciudades y otras entidades geográficas no son exactamente las mismas con las que se entrenó el modelo capaz de predecir los ingresos en la sección 7.1. Por ejemplo, si en la fase de preprocesado

del *pipeline* el codificador utilizado ha sido entrenado para reconocer el país Estados Unidos como *united states* y el modelo generado se utiliza para predecir una instancia etiquetada con el *tag USA* (como es el caso), el modelo entenderá que dicha empresa no tiene porque pertenecer a Estados Unidos. Lo mismo pasa con otros países, pero afortunadamente no con muchos. Para solucionar esto se incorporará un conjunto de reglas en la fase de preprocesado (justo antes de la codificación) que buscará unas etiquetas determinadas en el dataset que contiene las empresas de las que se quiere predecir algo para sustituirlas por los valores adecuados, es decir, sus etiquetas correspondientes que sí entenderá el modelo. Ahora bien, como se mostró en la sección 7.1.4, el modelo reconoce más de 15000 *tags* geográficos distintos, y en el dataset objetivo, que tiene cientos de miles de empresas distintas repartidas por todo el mundo, existirán muchas más. Por lo tanto hay que plantear una estrategia en cuanto a la selección de qué reglas se deben incluir. Para hacer un uso eficiente de los recursos, se analizarán cuáles son los *tags* más frecuentes en cada uno de los datasets, de tal manera que con corregir los 20 errores más frecuentes se reducirá en gran cuantía el número de falsos negativos en favor de más verdaderos positivos. Es decir, se conseguirá un sistema más exhaustivo.

Por otro lado, una vez decidido el modelo para estimar los ingresos (*ExtraTrees*), se entrenará con el dataset entero (sin dejar instancias para evaluar/testear) y se guardará en memoria para poder utilizarlo cuando se quiera sin necesidad de volver a entrenar. Para almacenarlo se serializarán al disco duro los objetos pertenecientes a los transformadores y estimadores. En total hay cuatro transformadores que se deben serializar:

- **MLBT**: *Multilabel Tags Binarizer*, encargado de binarizar un conjunto de *Tags*
- **MLBG**: *Multilabel GTags Binarizer*, encargado de binarizar un conjunto de *GTags*
- **MLBD**: *Multilabel Domains Binarizer*, encargado de binarizar un conjunto de *Domains*
- **Pipeline**: contiene un *pipeline* de dos pasos, en el primero hay un objeto *FeatureUnion* que selecciona todas las columnas que forman la entrada del modelo (preprocesadas previamente por los transformadores anteriores según el caso) y a continuación el objeto estimador *ExtraTrees* con la configuración deseada.

Para serializar cada transformador se utilizará la función *joblib.dump* y se cargará posteriormente mediante *joblib.load*, como viene explicado en el capítulo *Model Persistance* de la documentación de *Scikit-learn* [14b].

$Ratio_1$

- **inputs**: $tags, gtags, revenue, employees$

Igual que en caso de los ingresos, la integración del modelo elegido (*ExtraTrees*) para estimar $Ratio_1$ constará primero del reentrenamiento con todo el dataset (para aprovechar todo el conocimiento extraído de *Yahoo Finance*) y posteriormente se serializará el objeto del estimador a disco de la misma manera que con el regresor de ingresos.

$Ratio_2$

- **inputs**: $(tags)$

Este caso es similar al del WACC, ya que como se justificó debidamente en la sección 7.3, se utilizarán las medias sectoriales de Damodaran y por lo tanto la entrada a esta función serán exclusivamente las etiquetas de actividad empresarial.

Valoración, V

- **inputs**: $(SR, Revenue, Ratio_1, Ratio_2, WACC, g)$

Es la función que ejecuta en el orden adecuado todos los submodelos anteriores. Igual que en el caso de la determinación de la actividad empresarial y por los mismos motivos, existen dos versiones: una puntual y otra por lotes.

Por otro lado, como se ha comentado anteriormente, cada vez que se mejore el sistema extractor de etiquetas o se añada un número importante de empresas al dataset, será interesante que el modelo las utilice para generar valoraciones más precisas. Para ello sería necesario reentrenar todos los módulos, ya que sería posible que apareciesen nuevas relaciones o patrones entre los datos. De esta manera, se mantendrán bien organizados los distintos módulos, con el objetivo de que estén listos para actualizarse en cualquier momento.

8.2. Ejecución y análisis

Con todas las partes conectadas es el momento de probar el método de valoración creado. Eso es justo lo que se hará en esta sección. Se utilizará un dataset de 1M de empresas para ello.

Estructura de los Flujos de Caja

Antes de ejecutar la función de valoración merece la pena ejecutar de manera aislada uno de los *pipelines* más importantes de modelo: el de los flujos de caja libres. Una vez que se tienen implementadas las funcionalidades para estimar los *Revenue*, $Ratio_1$ y $Ratio_2$, ya se pueden calcular la estructura de los flujos de caja de las empresas del dataset objetivo. En la figura 8.1 se muestran los resultados obtenidos para las industrias más comunes mediante 3 histogramas superpuestos. A continuación se hacen algunos comentarios acerca de los resultados (sin ceñirse exclusivamente a las industrias de la figura 8.1).

- Llama la atención que el efectivo que las compañías son capaces de generar una vez que se han desprendido del dinero requerido para mantener o expandir sus negocios es de un cuarto de sus ingresos anuales en el mejor de los casos.
- Por otro lado, la diferencia entre los FCF y OCF es en general pequeña. Esto significa que la cantidad de inversión de las compañías en aumentar sus capacidades de generar beneficios (CAPEX) cada año es muy pequeña en comparación con los gastos en los que incurren como resultado de sus procesos operativos de negocio habituales (OPEX).
- Es relevante también el sector de sanidad (*healthcare*) por ser de los que menos flujos libres de caja tiene respecto a sus ingresos. No sólo eso sino que además en términos absolutos son menores que el CAPEX. Esto tiene sentido ya que la inversión en activos físicos y maquinas especializadas en este sector es muy importante.
- En el gráfico se han dibujado las desviaciones típicas de cada barra. De todas las industrias, la de computación en la nube es la que muestra unos ingresos más variables entre las distintas empresas. Se piensa que este hecho puede venir motivado porque hoy en día hay mucha competencia entre las empresas que utilizan esta tecnología y a muchos niveles, pero sobretodo porque las empresas que implementan esta tecnología pueden utilizar modelos de negocio variados (*SaaS*, *PaaS*, *IaaS*...).

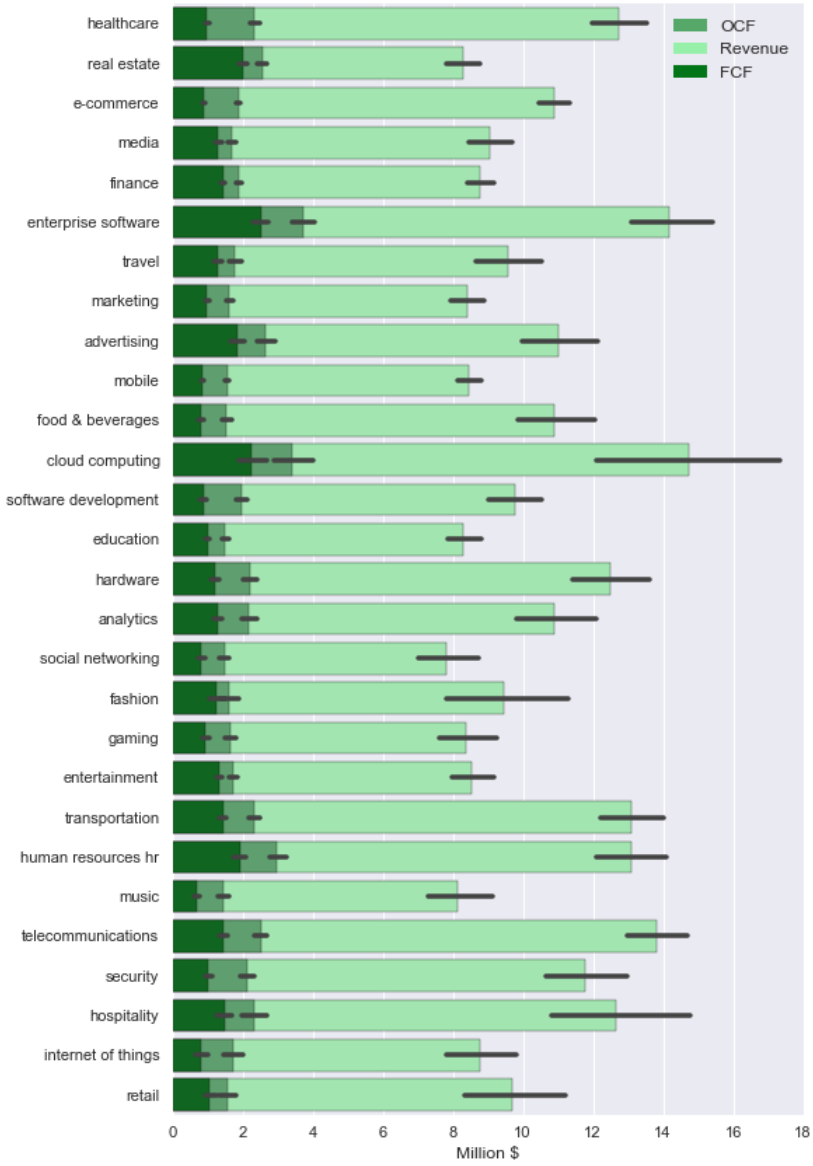


Figura 8.1: Ingresos (*Revenue*), flujos de caja operativos (OCF) y flujos libres de caja (FCF) de algunas de las industrias más relevantes

Distribución de las Valoraciones

A continuación se analizará el tipo de distribución que siguen las valoraciones otorgadas por el modelo a las empresas del dataset objetivo que se está utilizando. En primer lugar se ejecuta la función *pd.Series.describe()* de *Pandas*, siempre útil para observar de un vistazo rápido las claves

de cualquier distribución (se recuerda que las unidades son millones de dólares americanos).

```
count    998217.00000
mean         10.991503
std          16.925648
min           0.007667
25%           3.623312
50%           6.068987
75%          11.034084
max         496.575094
Name: V, dtype: float64
```

Con esto ya se pueden sacar algunas conclusiones. La valoración media ronda los 10 millones de dólares, y destaca que la máxima es de 496M. Dado que el percentil 75 es sólo de 11 unidades, si se dibujase en una misma gráfica la distribución completa quedaría una visualización poco reveladora ya que unos pocos datos con valores atípicos arruinarían la vista. Por este motivo, en la figura 8.2 se muestra la distribución de los valores junto con una estimación de la densidad del *kernel* sólo para aquellas valoraciones inferiores a 75M. Los resultados son estables, existiendo valoraciones de distintos órdenes de magnitud, desde miles de dólares hasta cientos de millones de dólares.

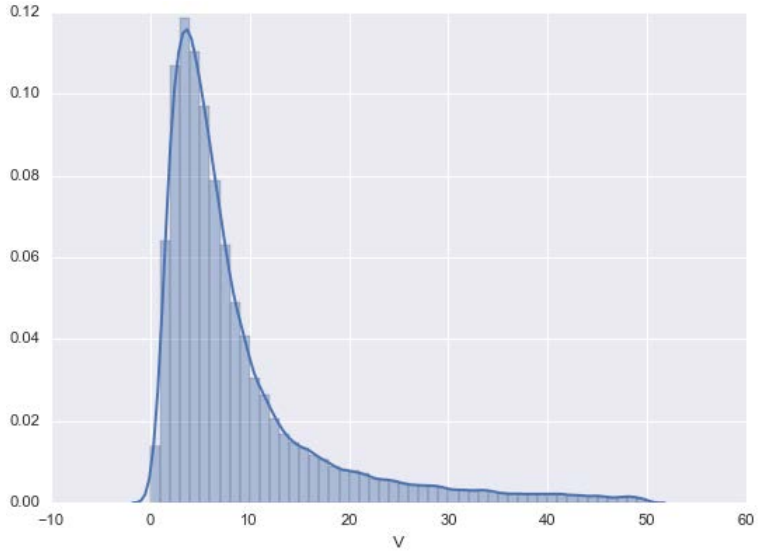


Figura 8.2: Distribución de las valoraciones

Atributos correlacionados con la Valoración

A continuación se analizará la correlación existente entre cada uno de los principales inputs de la fórmula, teniendo en cuenta la valoración. Ya se sabe el diseño de la fórmula, por lo que se espera verificar el efecto de cada componente. Para llevar a cabo el análisis se utilizará la figura 8.3 para mostrar él mapa de calor y la siguiente salida en la que se han calculado las correlaciones para la valoración:

```
Revenue          0.763455
OCF/Revenue      0.277422
SR               0.230881
g                0.150395
WACC            -0.222524
CAPEX/Revenue   -0.127997
Valuation        1.000000
Name: Valuation, dtype: float64
```

- Como era de esperar, los ingresos, el $Ratio_1$, las probabilidad de éxito y el crecimiento tienen una correlación positiva con la valoración.
- De todas ellas los ingresos es claramente el factor que más correlacionado está ($\rho = 0{,}78$).
- Por otro lado, el WACC y el $Ratio_2$ tienen correlación negativa con la valoración.

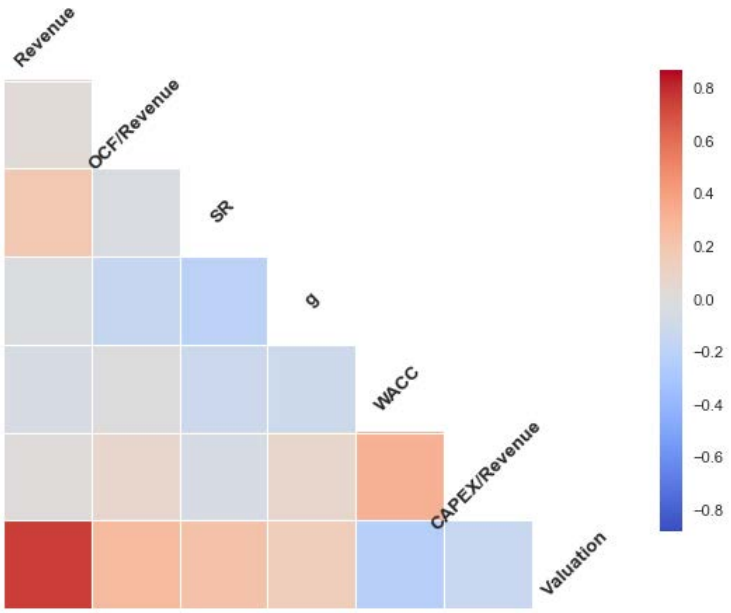


Figura 8.3: Mapa de calor de los inputs y V

Valoraciones según Localizaciones Geográficas

Para ello se ha dibujado en la figura 8.4 dos histograma con algo de ruido añadido para visualizar las valoraciones de las empresas en las ciudades y regiones con más empresas.

- En primer lugar, la mayoría de empresas del dataset tienen su sede en Estados Unidos y más concretamente en el área de la Bahía de San Francisco, donde se sabe que existe un denso ecosistema de *startups*.
- No parece que las compañías de unas localizaciones geográficas sean mayores que las de otras, pero si la densidad dentro de cada una de ellas. Por ejemplo, en la ciudad de Madrid hay muy pocas *startups* que superen los $50M.
- Dentro de cada localización geográfica parece que se respeta la distribución de valoraciones estudiada previamente.

Distribución de las Valoraciones según la Industria

En la figura 8.5 se ha intentado representar la distribución de la valoración atendiendo a la actividad de las empresas.

- Se verifica el hecho de que hay valoraciones muy dispares en todas las industrias. Esto es especialmente cierto en el sector financiero, lo que se explica por la importante transformación que esta sufriendo en los últimos años con motivo de la implantación de las nuevas tecnologías (*Fintech*). Existe una importante variabilidad porque tanto compañías como *startups* hacen apuestas distintas sobre qué tecnología liderará las próximas décadas el sector, como por ejemplo la presencia de las divisas virtuales.
- Hay algunas industrias cuyas valoraciones medias específicas se sitúan bastante por debajo de la media general, destacando el sector de la música con $3,6M, quizás porque se trata de un mercado con importantes barreras de entrada: por un lado, se podría hablar de oligopolio ya que el mercado, además de saturado, está repartido entre unas pocas compañías, y por otro todavía no se ha alcanzado un equilibrio legal en lo que a regulación de la piratería se refiere, lo cuál añade incertidumbre.

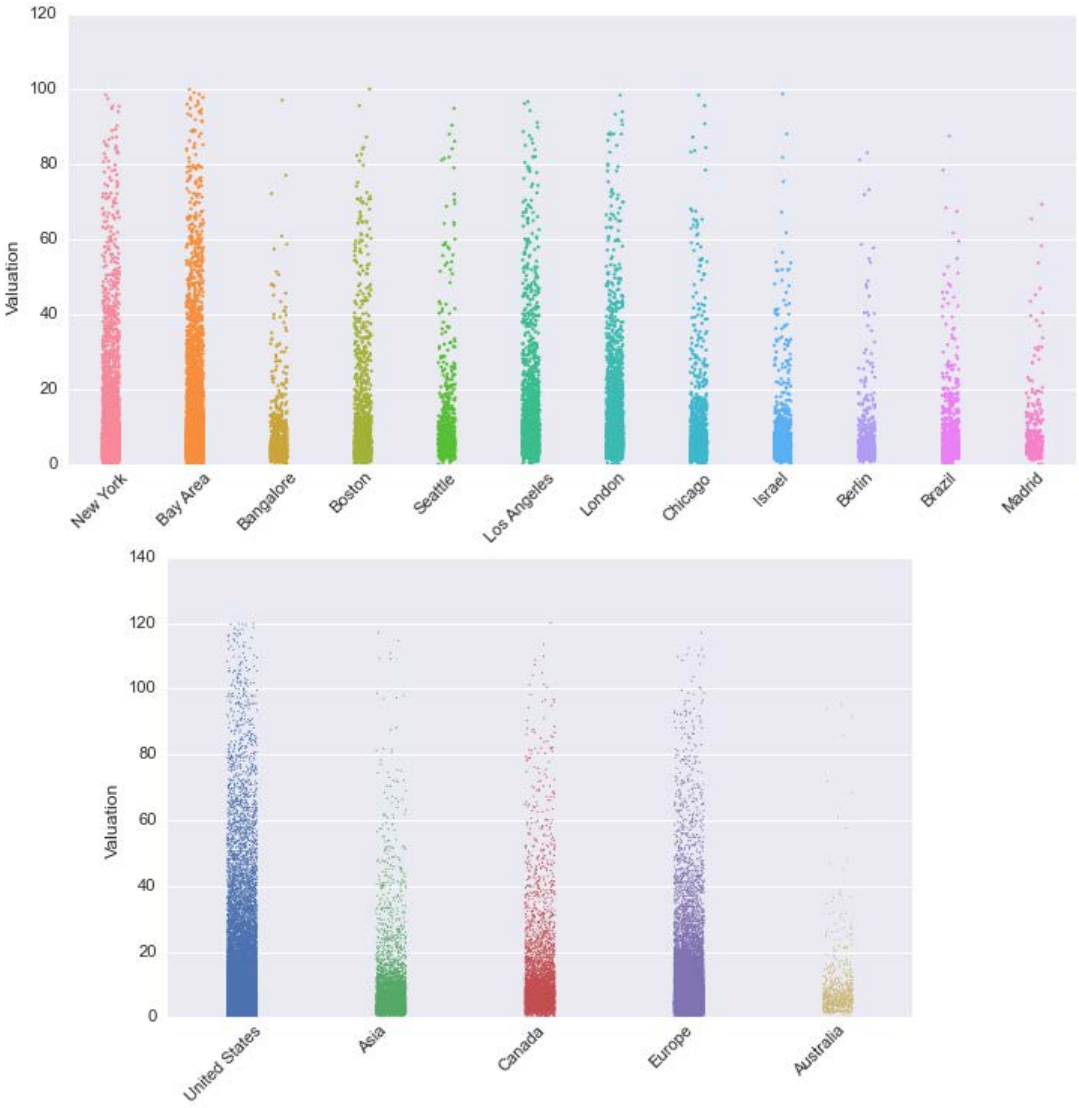


Figura 8.4: Valoraciones atendiendo a localizaciones geográficas

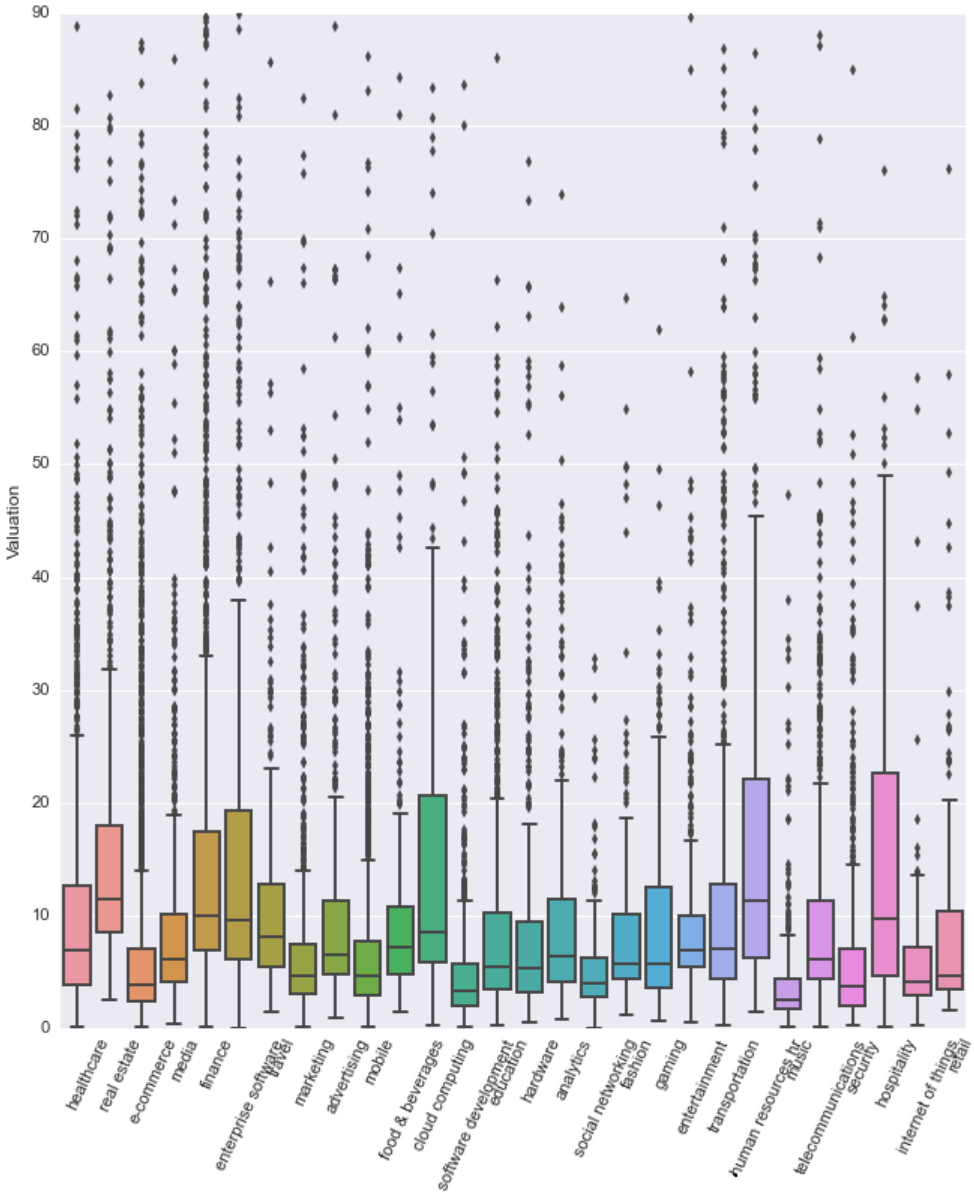


Figura 8.5: Distribución de las valoraciones según la industria

Correlación entre Ingresos y Valoración según el Modelo de Negocio

Hasta este punto está más que asegurado que el nivel de ingresos de una empresa es muy importante para obtener una buena valoración. Por ello, resulta de especial interés analizar qué factores alteran esta relación (*Revenue-Valuation*). En la figura 8.6 se han estimado varios modelos de regresión para visualizar como es la relación entre los ingresos de una empresa y su valoración dependiendo del modelo de negocio de esta. No se conoce el modelo de negocio de todo el dataset, por lo que se ha extraído una muestra de la que sí (400K ejemplos).

- Los modelos de negocio más usados por las empresas son B2C (*Business to Consumer*) y B2B (*Business to Business*).
- El modelo de negocio que más depende de un alto nivel de ingresos para ser valorado positivamente es el *Affiliate Marketing*. Tiene sentido porque en este paradigma, es en el momento en el que la empresa genera ventas o tráfico en su portal web cuando sus *partners* le pagan por haber alojado su publicidad en la página. El segundo modelo de negocio cuya valoración está más positivamente correlacionada con los ingresos que genera es SaaS o *Software as a Service*. Este tipo de negocios sólo son rentables cuando se acoplan a un modelo estratégico de ingresos. Además, es crucial para los mismos lo que se conoce como '*Revenue Recognition*': no importa cuanta caja llegue a la cuenta por parte del cliente, no se deben contabilizar los ingresos hasta que se haya entregado el producto.
- En general existe una relación positiva importante entre todos los modelos de negocio, salvo en el modelo de planificación de eventos. Esto puede deberse al hecho de que en este contexto existan otros factores que cobren más importancia respecto al nivel de ingresos, como la imagen de marca o cantidad y calidad de contactos. Al fin y al cabo una empresa que organiza eventos será mejor valorada cuanto más desarrolle su habilidad de atraer y conectar organizaciones y llamar la atención de la prensa, factores que si bien auguran importantes ingresos en el futuro, no tienen por qué hacerlo en el momento presente.

Correlación entre Número de Empleados y Valoración en función de la edad de la empresa

En la figura 7.10 se observó una estimación de la densidad del *kernel* de los ingresos respecto al número de empleados. Una correlación de $\rho = 0{,}56$ identificó que existía una relación importante entre ambos. En el apartado anterior se ha estudiado qué factores afectan a la relevancia de los ingresos para que una compañía sea valorada positivamente. Es el objetivo de este apartado analizar qué factores afectan a la relación entre la valoración y el número de empleados (que por transitividad se intuye positiva). Concretamente, se ha pensado que sería interesante elegir como factor la edad de la empresa. Por tanto, la pregunta a la que se intentará dar respuesta en este apartado es: ¿a qué edad es más importante para que una empresa sea valorada positivamente tener un alto número de empleados? En la figura 8.7 se intenta dar respuesta a esta pregunta. Para ello, antes de generar la visualización se creó una columna nueva en la que se asignó a cada empresa una categoría en función de su edad: 1, 2, 3, 4 o más de 4 años. Después, se entrenó un modelo de regresión lineal independiente para cada categoría.

- El primer dato que se desprende de su análisis es que la relación entre empleados y valoración se va haciendo más positiva según la empresa va cumpliendo años.

- Además, se observa también que estos aumentos se producen con menos magnitud en las empresas más maduras. Si se asume que el tamaño de las empresas aumenta con su edad, es razonable interpretar este hecho como que la contribución marginal a la valoración de contratar un empleado más disminuye según la empresa se va haciendo mayor, lo cuál tiene sentido.

- Aprovechando que en la figura 8.7 hay menos líneas que en la 8.6, se ha sombreado cada uno de los modelos indicando la fuerza de la correlación sin perjudicar a la claridad de la visualización. Se observa como dicha fuerza y por tanto la robustez del regresor es menor cuanto más joven es la empresa. Es decir, el número de empleados predice peor la valoración en *startups* que en compañías maduras. Esto es coherente porque en los inicios del emprendimiento, los factores que más determinan una buena valoración son la identificación de una necesidad del mercado, gestión adecuada de los fondos de maniobra, capacidad para pivotar y **calidad** del equipo, no **cantidad**.

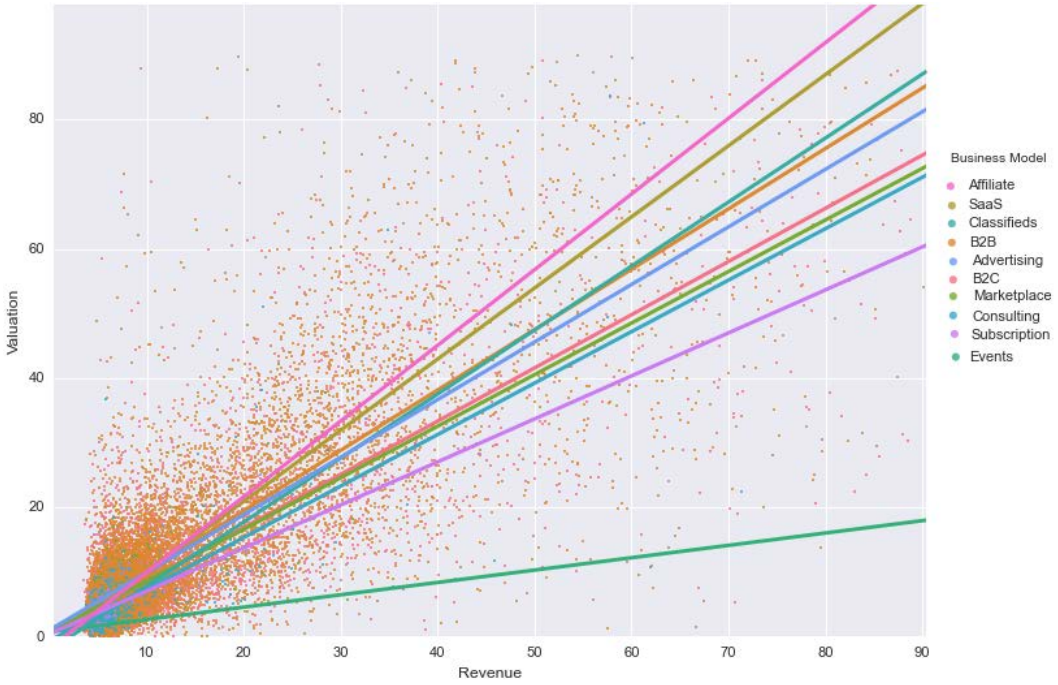


Figura 8.6: Correlación entre Ingresos y Valoración según el modelo de negocio

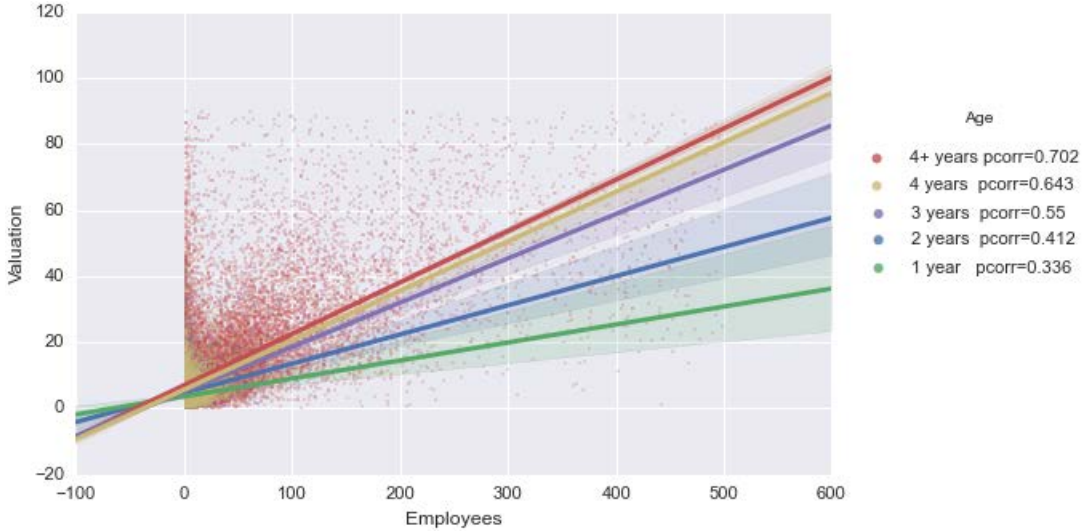


Figura 8.7: Correlación entre número de empleados y Valoración según la edad de la empresa

Capítulo 9

Conclusions and Future Work

9.1. Conclusions

Keeping in mind the objectives of the project that were stated in section 1.3, the author is satisfied with the results obtained. In order to write a good conclusion that puts together all of them, in the following paragraphs it will be explained to what extent has each objective been fulfilled:

1. **Measures utility**: This was the main goal of the project because it allows to rank companies according to an interesting criteria (a precious feature for the clients). This objective has been accomplished. Nevertheless, there is certainly room for improvement, since the precision of the models it's not perfect.

2. **Monetary value**: The capability of giving valuation in monetary units as output was the second most relevant objective. It was achieved thanks to the underlying basis of the intrinsic value estimated with the Discounted Cash Flows Model. In the same terms that the first objective, there is still scope to improve the monetary value.

3. **Scalable**: Thanks to the appropriate selection of the toolset (described throughout chapter 4), the resulting model's performance is 100 % on demand & flexible: it can be applied to specific startups on a case by case basis and also to large datasets in batches. In both versions, the performance is pretty good.

4. **Integrated system**: The resulting valuation method is composed by many modules. All of them have been successfully unified into one single pipeline process. Therefore, the obtained functionality is able to run in the server and is ready to become part of the business processes.

5. **Industry insights**: Some of the submodels were useful to visually analyse company's data and provided industries' comprehension. Figures A.2, 8.7, 8.6, 7.20, 8.2, A.4, 7.7, 7.10, 6.3, A.1, 6.1, A.3, 7.11, 8.3, 8.5, 7.12, 7.9, 7.19, 7.18, 8.1, 7.6, 7.8 and 7.8 represent some of the knowledge extracted.

6. **Learning path**: The knowledge related to Data Science acquired through the development of this university project are unmeasurable. The concepts learnt are being applied into other

projects related with Machine Learning at Global Incubator. Moreover, the author has lived the data scientist reality: most of the time is usually spent collecting and cleaning up data. As people often say: 'the best way of learning Machine Learning is getting your hands dirty with it'. On the other hand, one of the most valuable learning assets obtained is the ability of taking advantage of the software community (`www.stackoverflow.com` and `www.github.com`). Thousands of problems, more than people usually imagine, have already been resolved by others and the solutions are online open to the public. Spending some time searching on this communities, which get bigger each day, and not 'reinventing the wheel' can save a lot of a developerś time and a company's money.

9.2. Future Work

Improve other Global Incubator's projects

During his stay in Global Incubator, many of the projects that the author has been involved with have required the implementation of web API's[1] with SOAP technology[2]. It would be a good idea to combine those skills with this project and create a new API where clients were able to compute valuations for their desired companies in an easy, fast and flexible way.

On the other hand, the author has also developed a startup recommender system where one of the user's activities is to make groups of companies. On one hand, the system learns in an unsupervised fashion which companies are similar to each other measuring different types of distances between them (each distance takes into account different sets of features that are customizable by the user). On the other hand, the system learns which types of startups are added to which types of groups by each type of user, without them being necessarily similar (supervised learning), and then use the learnt patterns to suggest users to add new companies to their groups. The valuation method could be used here as a new feature to take into account when making recommendations. There are many interesting options: it would be very useful to rank any recommended output according to the valuation. Also, it would be a good idea to use the valuation in the finding patterns and similarities process.

Deep Learning

The Machine Learning models applied in this project are wide used in the industry. However, Deep Learning models have become the trending field of Data Science because of their remarkable achievements[3]. As it was said in section 4.1.2, Scikit-learn, the Machine Learning library used in this project, does not support these state of the art algorithms. However, there are other tools specifically designed for training Deep Learning models which due to the hard work of the community are now in mature state. One of them is Keras [Cho15] and the author is already using it in other projects at Global Incubator (generating abstractive news summaries), so it would be worth to test the performance of some Neural Net models with the problems that have been tackled in this project.

One of the differences between the models trained in this project and Deep Learning algorithms is that the latter usually outperform the former when the dataset is bigger. Taking also into account

[1] Application Programming Interface for web applications

[2] Simple Object Access Protocol is a protocol specification for exchanging structured information in the implementation of web services in computer networks

[3] `http://www.kdnuggets.com/2016/01/deep-learning-overhyped.html`

that the regular neural nets are very CPU expensive, this means that one cannot deal seriously with Deep Learning without large computing resources. The server used in this project (40 CPUs and 256gb RAM) is quite powerful and could support some Deep Learning training processes, however, it would be much more efficient (both in terms of speed and electric bills) to use a Graphic Processor Unit based architecture. This is the kind of information processing machinery that is currently used for that matter[4]. Before spending the money on one of this powerful processor units it is worth to consider renting. There already are many cloud computing corporations from which to rent these kind of computer resources at spot price, which sometimes is very cheap. One important disadvantage is the fact that when spot price changes, the rented computing instance gets shutdown with the risk of loosing the work. Note that it would be feasible to deal with this drawback, since it is possible to save the model back after each training iteration. Although a thorough analysys will have to take place before making the buy-vs-rent decision, it looks like it would be a good strategy to first rent the resources for validating the models and finally invest in building a GPU architecture for production.

More and better data

Global Incubator is improving the quality and quantity of its startups dataset through different paths:

- The sources from which crawlers are scrapping the data are increasing, which will result in more startups to analyze and train better models.
- In order to improve the quality of the dataset, it will be useful to increase the number of features that these scrappers gather. One example could be to collect social indicators of more social networks. Specially talking about startups, a more elaborated 'social traction' indicator will probably correlate considerably with its valuation.
- Global Incubator is negotiating with organizations that can provide more detailed information of each company. If some agreement is finally reached, the possible models to train would be much more diverse. As a matter of fact, it would be quite interesting to fit a RNN model[5] that tries to predict the future cash flows of a company based on its historic revenue.
- Related with the last point, it will be very useful to get financial data from startups that are doing well, but also from startups that are bad examples to follow. Hence, the feature distributions of the companies would be closer the actual population distribution, which translates to reduce the bias of the estimators fitted.

Startups' WACC adjustment

As it was said in section 6.2, young companies of one sector tend to covariate more drastically with the market than big enterprises (of the same sector). As a matter of fact, some studies [Coc01] [MH99] acknowledge that late funding stages are less risky. Startups have progresively lower volatility has they move forward, and therefore lower arithmetic average returns. If a big company (like the ones Damodaran uses in his datasets[16a]) has a β of lets say 1.0 it is likely that a startaup that

[4] `http://timdettmers.com/2015/03/09/deep-learning-hardware-guide/`

[5] Recursive neural net used to fit estimators with the capability of having an internal state which allows it to exhibit dynamic temporal behavior.

belong to the same sector will have a noticeable higher β, i.e 1.5 or 2.0. This mean that startup ecosystems tend to be quite more volatile and therefore investors such as Venture Capitals will require higher returns on equity. It is worth to note that this is not because of the risk of failure of the startup but for the uncertainity that surrounds it. Venture Capitals are primarly known for the high risk that they assume by investing in smaller and early-stage companies, and that is why their investments usually fail. On the other hand, from time to time they succeed and sometimes they find unicorns[6]. What is interesting about their strategy is that they usually earn more profit in average than those investors that put their money in less risky positions with less returns. This contradicts the CAPM, which states that at least theoretically it doesn't matter what investments are made by an agent, it will always get the same relation between variance and average return (Sharpe ratio[7] equilibrium). Of course, this statement differs from reality.

Now, in order to achieve a better adjustment of the startups' WACC it will be useful to set Damodaran's as lowerbound, since it is reasonable to think that nobody will require lower returns from a startup than from an enterprise that belongs to the same sector. From that point, a strategy has to be taken to increase the returns of riskier startups. [MH99] is a paper that analyzes the risk and return of venture capital investments, where *selection bias* is the central hurdle. For example, one of the problems resides in the fact that many firms in its sample remain privat and the ones that go public (IPO[8]) usually have done a good job and have experienced good returns. Therefore, a return to IPO analysis will lead to biased results, since it would only measure only the 'winners'. [Coc01] is another study that had to overcome the same problem: 'There is a paucity of reliable information about the returns from venture capital investing. Institutional venture capital firms do not generally make public any information about their actual returns and hence most of the research has been restricted to the performance of the small minority of venture capital firms that are publicly quoted'. From this studies it is extracted that venture capital returns on equity have most often been in the teens, with occasional periods in the 20 % to 30 % range and rare spikes above 30 % (average of 25 %). Damodaran's average market WACC is 6,3 %, 4 times lower. Therefore, a good strategy for this project would to multiply the already computed WACC from matched Damodaran's industrial averages (section 6.2) by $k \in [1, 4]$. Figure 9.2 show the that the k function chosen has been built makeing some linear transformations to a regular sigmoud function.

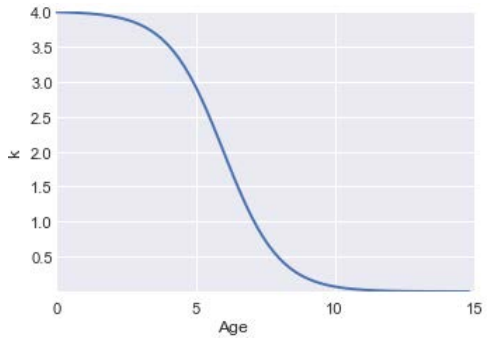


Figura 9.1: k function

[6]Startup company valued at over $1 billion USD.
[7]It characterizes how well the return of an asset compensates the investor for the risk taken.
[8]An initial public offering is the first sale of stock by a private company to the public.

Note also that k is a function of the company's age, since younger startups tend to be more volatile. Moreover, its sigmoid shape its inspired on the nature of the classic startup cycle shown in figure 9.2. This last image also suggest that taking the stage of investment of the company into account would be a good idea since it may probably lead to more precise results. However, the amount of companies with known age in the dataset is much higher than those with stage of funding. This is not a data collection issue. Instead, it happens that sometimes private equity funding rounds are undisclosed. Instead of using the k function, another strategy to follow would be to take both upper and lower bounds and compute confidence intervals.

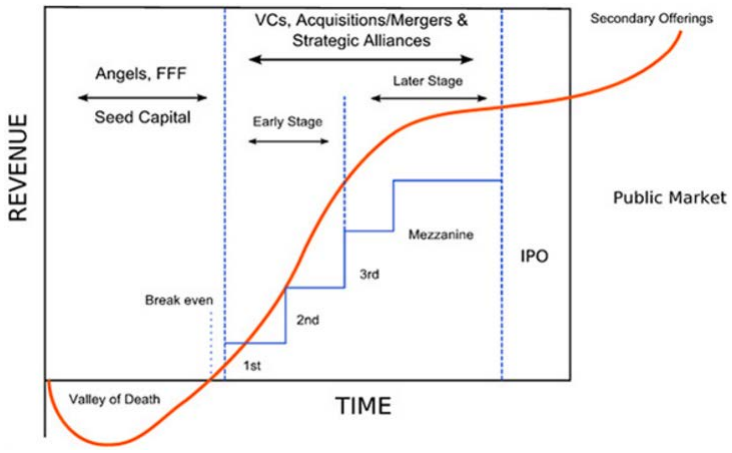


Figura 9.2: Startup cycle. Source: *Wikipedia*

In fact, there already exist some studies that analyse returns of VCs according to a startup stage [Dam09]:

Venture Capital Target Rates of Return – Stage in Life Cycle	
Stage of development	Typical target rates of return
Start up	50-70 %
First stage	40-60 %
Second stage	35-50 %
Bridge/IPO	25-35 %

Cuadro 9.1: VCs required return on equity as a function of startup stage of development

It is noticeable that those returns are much higher than the WACC estimated in this project. It is very important to realize that the returns in table 9.1 are far away from the ones estimated here because they reflect the idiosincratic risk of failure of the startup and therefore don not represent the WACC. In order to do so, that risk must be removed from the discount rate. Then, it should be used to adjust the expected cash flows in the case a valuation needs to be performed (just like it has been explained in this project).

Rule Extractor Improvements

In the case of the Rule Expert System that is in charge of determine economic activity of a company (sección 5.2) there some improvements that will take place in following iterations of the project:

- Through constantly debugging the rules in the `csv` files that are used to match industry tree nodes as soon as any bugs are detected. Through the creation of new rule files with more search keywords in order to being able to get a better picture of the economic activity of companies:
- Another interest optimization would be to take a look at the Python's Natural Language Toolkit (NLTK). This library provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet[9], along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries. This has the potential of improve the quality of economic and location tags.

[9] `https://wordnet.princeton.edu/`

Referencias

[Smi76] Adam Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations*. W. Strahan y T. Cadell, London, 1776. URL: http://www.ibiblio.org/ml/libri/s/SmithA_WealthNations_p.pdf.

[Sch08] Joseph Alois Schumpeter. *The theory of Economic Development: An Inquiry into Profits, Capital, Credit, Interest and The Business Cycle*. research report. Harvard Economic Studies, 1934 (2008).

[37] *Standard Industrial Classification*. Gobierno de Estados Unidos. 1937. URL: http://www.osha.gov/pls/imis/sic_manual.html.

[Ros56] Murray Rosenblatt. *Remarks on Some Nonparametric Estimates of a Density Function*. research report. University of Chicago, 1956. URL: http://projecteuclid.org/euclid.aoms/1177728190.

[Efr79] Bradley Efron. *Bootstrap Methods: Another Look at the Jacknife*. research report. Standford University, 1979. URL: http://stat.cmu.edu/~fienberg/Statistics36-756/Efron1979.pdf.

[Ken83] John T. Kent. *Information Gain and a General Measure of Correlation*. research report. Biometrika, 1983. URL: http://jstor.org/stable/2335954?seq=1#page_scan_tab_contents.

[MF90] John McCarthy y Edward Feigenbaum. *In Memorian Arthur Samuel: Pioner in Machine Learning*. research report. AI Magazine, 1990. URL: http://aaai.org/ojs/index.php/aimagazine/article/view/840/758.

[Fou91a] Python Software Foundation, ed. *EAFP (Python 2 Glossary)*. 1991. URL: http://docs.python.org/2/glossary.html#term-eafp.

[Fou91b] Python Software Foundation, ed. *LBYL (Python 2 Glossary)*. 1991. URL: http://docs.python.org/2/glossary.html#term-lbyl.

[91] *Zen of Python*. aaaaa. 1991. URL: http://en.wikipedia.org/wiki/Zen_of_Python.

[Kir92] Alan P. Kirman. *Whom or What Does the Representative Individual Represent?* Journal of Economic Perspectives, 1992. URL: http://econ2.econ.iastate.edu/tesfatsi/WhomOrWhatDoesRepIndRepresent.AKirman1992.pdf.

[Bre94] Leo Breiman. *Bagging Predictors*. research report. University of California at Berkeley, 1994. URL: http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf.

[Koh95] Ron Kohavi. *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*. 1995, págs. 1137-1143.

[Dru97] Harris Drucker. *Improving Regressors using Boosting Techniques*. research report. Monmouth University, 1997. URL: http://professordrucker.com/Pubications/ImprovingRegressorsUsingBoostingTechniques.pdf.

[97] *North American Industry Classification System*. Gobierno de Estados Unidos. 1997. URL: http://naics.com.

[Hol98] Randall G. Holcombe. *Entrepreneurship and economic growth*. research report. The Quarterly journal of Austrian Economics, 1998. URL: http://mises.org/system/tdf/qjae1_2_3.pdf?file=1&type=document.

[MN98] Andrew McCallum y Kamal Nigam. *A Comparison of Event Models for Naive Bayes Text Classification*. research report. Carnegie Mellon University, 1998. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.9324&rep=rep1&type=pdf.

[98] *Write Once, Run Anywhere: Why It Matters*. interhack. 1998. URL: http://interhack.net/people/cmcurtin/rants/write-once-run-anywhere/write-once-run-anywhere.pdf.

[GHO99] Gene H. Golub, Christian Hansen y Dianne P. O'Leary. *Tikhonov Regularization and Total Least Squares*. research report. Society for Industrial y Applied Mathematics, 1999. URL: http://drum.lib.umd.edu/bitstream/handle/1903/913/CS-TR-3829.pdf.

[MH99] Colin M. Mason y Richard T. Harrison. *Is it worth it? The rates of return from informal venture capital investments*. Journal of Business Venturing, 1999. URL: http://sciencedirect.com/science/article/pii/S0883902600000604.

[Ram99] Juan Ramos. *Using TF-IDF to Determine Word Relevance in Document Queries*. research report. Rutgers University, 1999. URL: http://www.cs.rutgers.edu/~mlittman/courses/ml03/iCML03/papers/ramos.pdf.

[C00] Shearer C. *The CRISP-DM model: the new blueprint for data mining*. research report. Journal of Data Warehousing, 2000. URL: http://mineracaodedados.files.wordpress.com/2012/04/the-crisp-dm-model-the-new-blueprint-for-data-mining-shearer-colin.pdf.

[Coc01] John H. Cochrane. *The Risk and Return of Venture Capital*. Anderson School of Management, 2001. URL: http://escholarship.org/uc/item/7qm9h594.

[Dam01] Aswath Damodaran. *The Dark Side of Valuation: Valuing difficult-to-value companies*. research report. Stern School of Business, New York University, 2001. URL: http://people.stern.nyu.edu/adamodar/pdfiles/country/darkside.pdf.

[Fri01] Jerome H. Friedman. *Greedy Function Approximation: A Gradient Boosting Machine*. research report. Institute of Mathematical Statistics, 2001. URL: http://luthuli.cs.uiuc.edu/~daf/courses/optimization/papers/2699986.pdf.

[MMRTS01] Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda y Bernhard Schölkopf. *An Introduction to Kernel-Based Learning Algorithms*. IEEE Transactions on Neural Networks, 2001. URL: http://home.eng.iastate.edu/~julied/classes/ee547/Handouts/Sholkoffreview.pdf.

[Nav01] Gonzalo Navaroo. *A Guided Tour to Approximate String Matching*. research report. Universidad de Chile, 2001. URL: http://repositorio.uchile.cl/bitstream/handle/2250/126168/Navarro_Gonzalo_Guided_tour.pdf.

[01] *Statistics and Probability Dictionary: Coefficient of Determination*. Stat Trek. 2001. URL: http://stattrek.com/statistics/dictionary.aspx?definition=coefficient_of_determination.

[02] *Matplotlib*. 2002. URL: http://matplotlib.org/.

[ZH03] Hui Zou y Trevor Hastie. *Regularization and variable selection via the elastic net*. Standford University, 2003. URL: http://web.stanford.edu/~hastie/Papers/B67.2%5C%20(2005)%5C%20301-320%5C%20Zou%5C%20&%5C%20Hastie.pdf.

[04] *Selenium*. ThoughtWorks. 2004. URL: http://seleniumhq.org.

[GEW05] P. Geurts, D. Ernst. y L. Wehenkel. *Extremely randomized trees*. research report. Machine Learning, 2005.

[CDKSS06] K. Crammer, O. Dekel, J. Keshat, S. Shalev-Shwartz e Y. Singer. *Online Passive-Aggressive Algorithms*. research report. Journal of Machine Learning Research, 2006. URL: http://jmlr.csail.mit.edu/papers/volume7/crammer06a/crammer06a.pdf.

[Fer07] Pablo Fernández. *Company Valuation Methods. The Most Common Errors in Valuations*. research report. IESE Business School-University of Navarra, 2007. URL: http://iese.edu/research/pdfs/di-0449-e.pdf.

[HTF08] Trevor Hastie, Robert Tibshirani y Jerome Friedman. *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*. Standford University, 2008. URL: http://web.stanford.edu/~hastie/local.ftp/Springer/OLD/ESLII_print4.pdf.

[Kab08] S. I. Kabanikhin. *Definitions and examples of inverse and ill-posed problems*. research report. Gruyter, 2008.

[08a] *Lean Startup Methodology*. The Lean Startup. 2008. URL: http://theleanstartup.com/principles.

[Ng08] Andrew Ng. *Machine Learning*. Coursera y Standford University, 2008. Cap. 7.1 The Problem of Overfitting. URL: http://class.coursera.org/ml-005/lecture/preview.

[08b] *Python Data Analysis Library*. Community. 2008. URL: http://pandas.pydata.org/.

[08c] *Scrapy*. Scrapinghub, Ltd. 2008. URL: http://scrapy.org.

[Ben09] Yoshua Bengio. *Learning Deep Architectures for AI*. research report. Foundations y Trends, 2009. URL: http://sanghv.com/download/soft/machine%5C%20learning,%5C%20artificial%5C%20intelligence,%5C%20mathematics%5C%20ebooks/ML/learning%5C%20deep%5C%20architectures%5C%20for%5C%20AI%5C%20%5C%282009%5C%29.pdf.

[Dam09] Aswath Damodaran. *Valuing Young, Start-up and Growth Companies: Estimation Issues and Valuation Challenges*. Stern School of Business, New York University, 2009. URL: http://people.stern.nyu.edu/adamodar/pdfiles/papers/younggrowth.pdf.

[MMS09] Óscar Marbán, Gonzalo Mariscal y Javier Segovia. *A Data Mining & Knowledge Discovery Process Model*. research report. Universidad Politécnica de Madrid y Universidad Europea de Madrid, 2009. URL: http://cdn.intechopen.com/pdfs/5937/InTech-A_data_mining_amp_knowledge_discovery_process_model.pdf.

[Bea10] David Beazley. *Understanding the Python GIL*. 2010. URL: http://www.dabeaz.com/python/UnderstandingGIL.pdf.

[Bot10] Léon Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent*. research report. 2010.

[10] *Business Model Generation*. 2010.

[CT10] G. C. Cawley y N. L. C. Talbot. *Over-fitting in model selection and subsequent selection bias in performance evaluation*. research report. Journal of Machine Learning Research, 2010. URL: http://jmlr.csail.mit.edu/papers/volume11/cawley10a/cawley10a.pdf.

[ERK10] Michael D. Ekstrand, John T. Riedl y Joseph A. Konstan. *Collaborative Filtering Recommender Systems*. research report. Foundations y Trends, 2010. URL: http://files.grouplens.org/papers/FnT%5C%20CF%5C%20Recsys%5C%20Survey.pdf.

[HG10] David W. Humphrey y Florian G. *Capital Expenditure Survey 2010*. ARC Strategies, 2010. URL: http://arcweb.com/featured-reports/Capital%5C%20Expenditure%5C%20Survey%5C%202010.pdf.

[RRSK10] Francesco Ricci, Lior Rokach, Brach Shapira y Paul B. Kantor. *Content-based Recommender Systems: State of the Art and Trends*. research report. Springer US, 2010. URL: http://link.springer.com/chapter/10.1007/978-0-387-85820-3_3.

[14a] *Feature importances with forests of tree*. Scikit-learn. 2010 - 2014. URL: http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html.

[14b] *Model Persistance*. Scikit-learn. 2010 - 2014. URL: http://scikit-learn.org/stable/modules/model_persistence.html.

[14c] *Multilabel ranking metrics*. Scikit-learn. 2010 - 2014. URL: http://scikit-learn.org/stable/modules/model_evaluation.html#multilabel-ranking-metrics.

[Sci14a] Scikit-Learn, ed. *Feature selection*. 2010-14. URL: http://scikit-learn.org/stable/modules/feature_selection.html.

[Sci14b] Scikit-Learn, ed. *Grid Search: Searching for estimator parameters*. 2010-14. URL: http://scikit-learn.org/stable/modules/grid_search.html#grid-search.

[Sci14c] Scikit-learn, ed. *MultinomialNB*. 2010-14. URL: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html.

[Sci14d] Scikit-learn, ed. *One-VS-the-Rest*. 2010-14. URL: http://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html.

[Fer11] Pablo Fernández. *WACC: Definición, Interpretaciones Equivocadas y Errores*. IESE Business School-University of Navarra, 2011. URL: http://iese.edu/research/pdfs/DI-0914.pdf.

[11] *FuzzyWuzzy: Fuzzy String Matching in Python*. ChairNerd. 2011. URL: http://github.com/seatgeek/fuzzywuzzy.

[git11] caesar0301 (github), ed. *Treelib*. 2011. URL: http://xiaming.me/treelib/.

[Nik11] M.S. Nikulin. *Loss function*. Encyclopedia of Mathematics. 2011. URL: http://www.encyclopediaofmath.org/index.php/Loss_function.

[Col12] Michael Collins. *Convergence Proof for the Perceptron Algorithm*. research report. Columbia University, 2012. URL: http://cs.columbia.edu/~mcollins/courses/6998-2012/notes/perc.converge.pdf.

[Cru12] Crummy, ed. *Beautiful Soup*. 2012. URL: http://crummy.com/software/BeautifulSoup/bs4/doc/.

[Dam12] Aswath Damodaran. *Valuation: Lecture Note Packet 1: Intrinsic Valuation*. research report. Stern School of Business, New York University, 2012. URL: http://people.stern.nyu.edu/adamodar/pdfiles/eqnotes/packet1b.pdf.

[Hen12] James Hendler. *Avoiding Another AI Winter*. research report. Rensselaer Polytechnic Institute, 2012. URL: http://web.archive.org/web/20120212012656/http://csdl2.computer.org/comp/mags/ex/2008/02/mex2008020002.pdf.

[MHDB12] Max Marmer, Bjoern Lasse Herrmann, Ertan Dogrultan y Ron Berman. *Startup Genome Report Extra on Premature Scaling*. technical report. Standford University y Startup Genome, 2012. URL: http://s3.amazonaws.com/startupcompass-public/StartupGenomeReport2_Why_Startups_Fail_v2.pdf.

[Tsa12] Chih-Fong Tsai. «Bag-of-Words Representation in Image Annotation: A Review». En: *International Scholarly Research Notices* 2012 (26 de ago. de 2012), pág. 19. DOI: 10.5402/2012/376804. URL: http://www.hindawi.com/journals/isrn/2012/376804/.

[YSLD12] Sheng Kung Michael Yi, Mark Steyvers, Michael D. Lee y Matthew J. Dry. *The Wisdom of the Crowd in Combinatorial Problems*. 2012. URL: http://onlinelibrary.wiley.com/doi/10.1111/j.1551-6709.2011.01223.x/full.

[13a] *Poll: Languages for analytics / data mining / data science*. KDnuggets. 2013. URL: http://www.kdnuggets.com/polls/2013/languages-analytics-data-mining-data-science.html.

[Ska13] Wolfgang Skala, ed. *Drawing Gantt Charts in Latex*. 2013. URL: http://mirror.unl.edu/ctan/graphics/pgf/contrib/pgfgantt/pgfgantt.pdf.

[13b] *The Four Steps to the Epiphany: Successful Strategies for Products that Win*. 2013.

[13c] *Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS*. Rackspace US, Inc. 2013. URL: http://support.rackspace.com/white-paper/understanding-the-cloud-computing-stack-saas-paas-iaas/.

[14d] *Capital expenditures net sales ratio: Industry Average Ranking*. EDIUNET, 2014. URL: http://industry.ediunet.jp/choice/529/?lang=en.

[LZCS14] Mu Li, Tong Zhang, Yuqiang Chen y Alexander J. Smola. *Efficient mini-batch training for stochastic optimization*. research report. Carnegie Mellon University, 2014. URL: http://dl.acm.org/citation.cfm?id=2623612.

[14e] *Poll: What main methodology are you using for your analytics, data mining, or data science projects?* KDnuggets. 2014. URL: http://kdnuggets.com/polls/2014/analytics-data-mining-data-science-methodology.html.

[14f] *Precission and Recall*. Wikimedia. 2014. URL: http://en.wikipedia.org/wiki/Precision_and_recall.

[14g] *S&P 500 LTM Capex by Sector*. Compustat y Goldman Sachs Global Investment Research, 2014. URL: http://businessinsider.com/capex-spending-by-industry-2014-2.

[Cho15] François Chollet. *Keras*. https://github.com/fchollet/keras. 2015.

[FMRR15] Robert W. Fairlie, Arnobio Morelix, E.J. Reedy y Joshua Russel. *The Kauffman Index Startup Activity*. research report. Ewing Marion Kauffman Foundation, 2015. URL: http://kauffman.org/~/media/kauffman_org/research%5C%20reports%5C%20and%5C%20covers/2015/05/kauffman_index_startup_activity_national_trends_2015.pdf.

[15a] *High Tech Startup Valuation Estimator*. Cayenne Consulting. 2015. URL: http://caycon.com/valuation.php.

[15b] *Top 20 Python Machine Learning Open Source Projects*. KDnuggets. 2015. URL: http://kdnuggets.com/2015/06/top-20-python-machine-learning-open-source-projects.html.

[CG16] Tianqi Chen y Carlos Guestrin. *XGBoost: A Scalable Tree Boosting System*. research report. University of Washington, 2016. URL: http://dmlc.cs.washington.edu/data/pdf/XGBoostArxiv.pdf.

[Dam16] Aswath Damodaran. *Capital Expenditures by Sector (US)*. 2016. URL: http://pages.stern.nyu.edu/~adamodar/New%5C_Home%5C_Page/datafile/capex.html.

[16a] *Financial current data*. Damodaran Online. 2016. URL: http://pages.stern.nyu.edu/~adamodar/New_Home_Page/datacurrent.html.

[16b] *Startup Business Failure Rate By Industry*. research report. Statistic Brain Research Institute, 2016. URL: http://statisticbrain.com/startup-failure-by-industry.

[Dama] Aswath Damodaran. *Models of Risk and Return*. research report. Stern School of Business, New York University. URL: http://people.stern.nyu.edu/adamodar/pdfiles/ovhds/ch3.pdf.

[Damb] Aswath Damodaran. *Relative Valuation*. Stern School of Business, New York University. URL: people.stern.nyu.edu/adamodar/pdfiles/country/relval.pdf.

[Goesd] William N. Goetzmann. *An Introduction to Investment Theory*. YALE School of Management, asdasd. URL: http://viking.som.yale.edu/will/finman540/classnotes/class2.html.

Apéndice A

Distribuciones estadísticas

A continuación se muestran por un lado las distribuciones de los atributos numéricos *Followers* y *Age* del dataset, y por otro la de las variables categóricas multi-etiqueta *GTags* y *Domains*, pertenecientes a la sección 7.1.

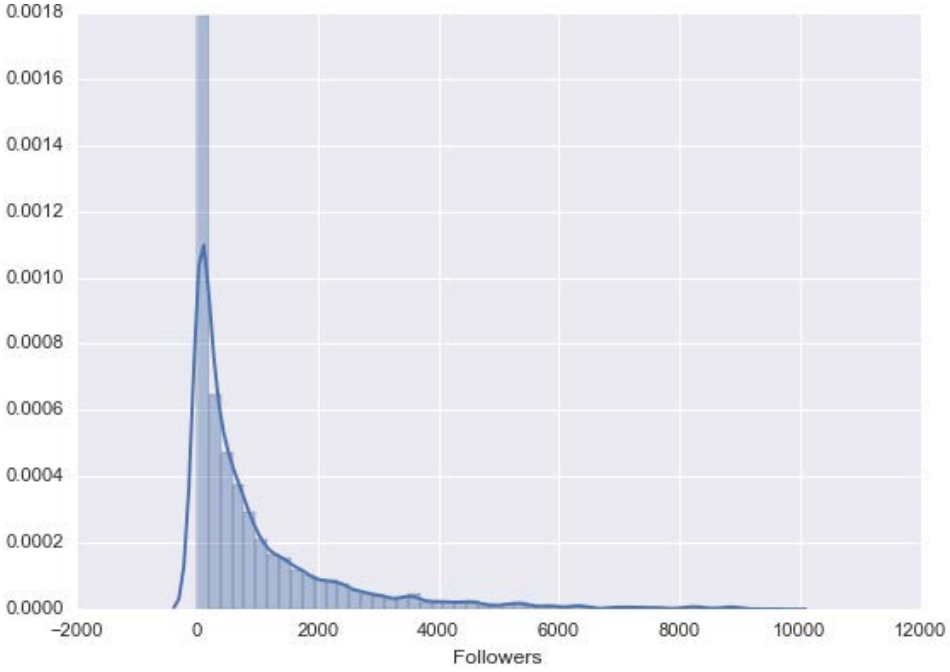


Figura A.1: Distribución de *Followers* en el dataset

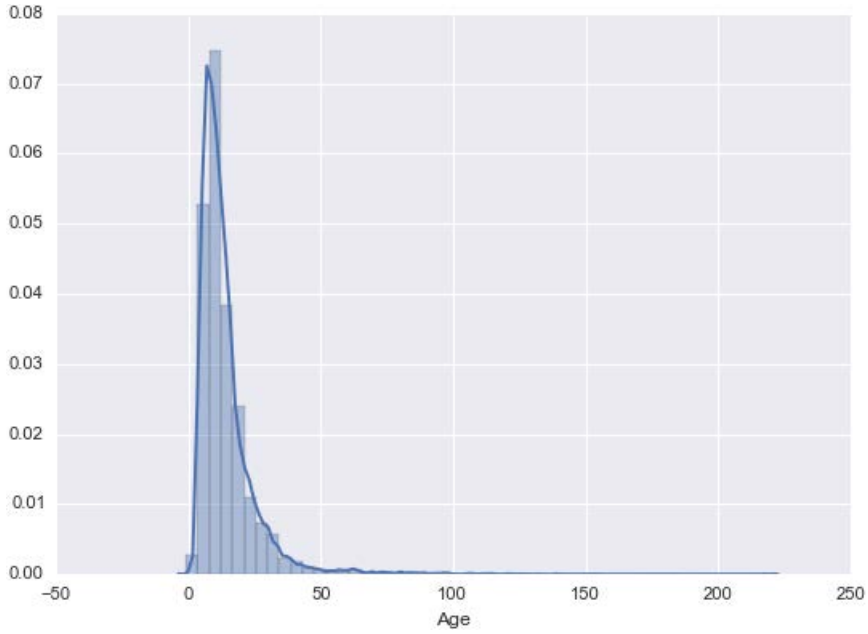


Figura A.2: Distribución de *Age* en el dataset

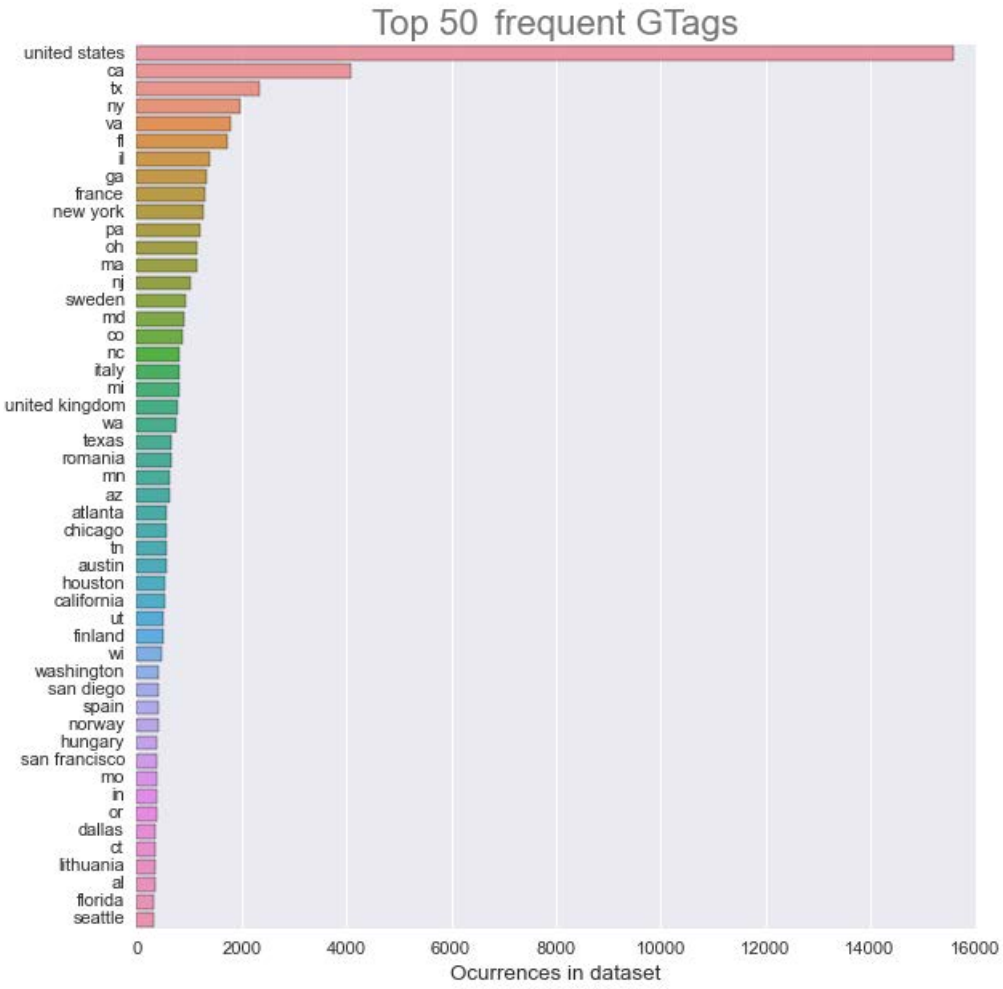


Figura A.3: GTags más frecuentes en el dataset de entre 4392 existentes

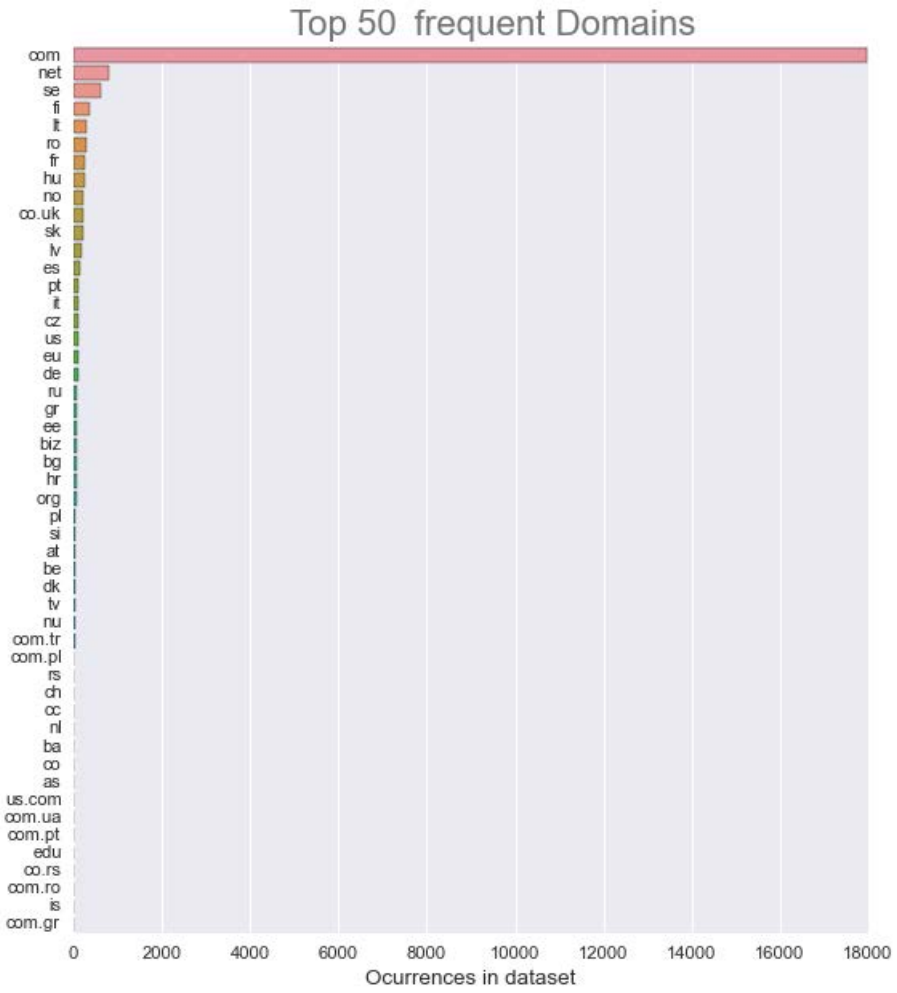


Figura A.4: Dominios más frecuentes en el dataset de entre 94 existentes

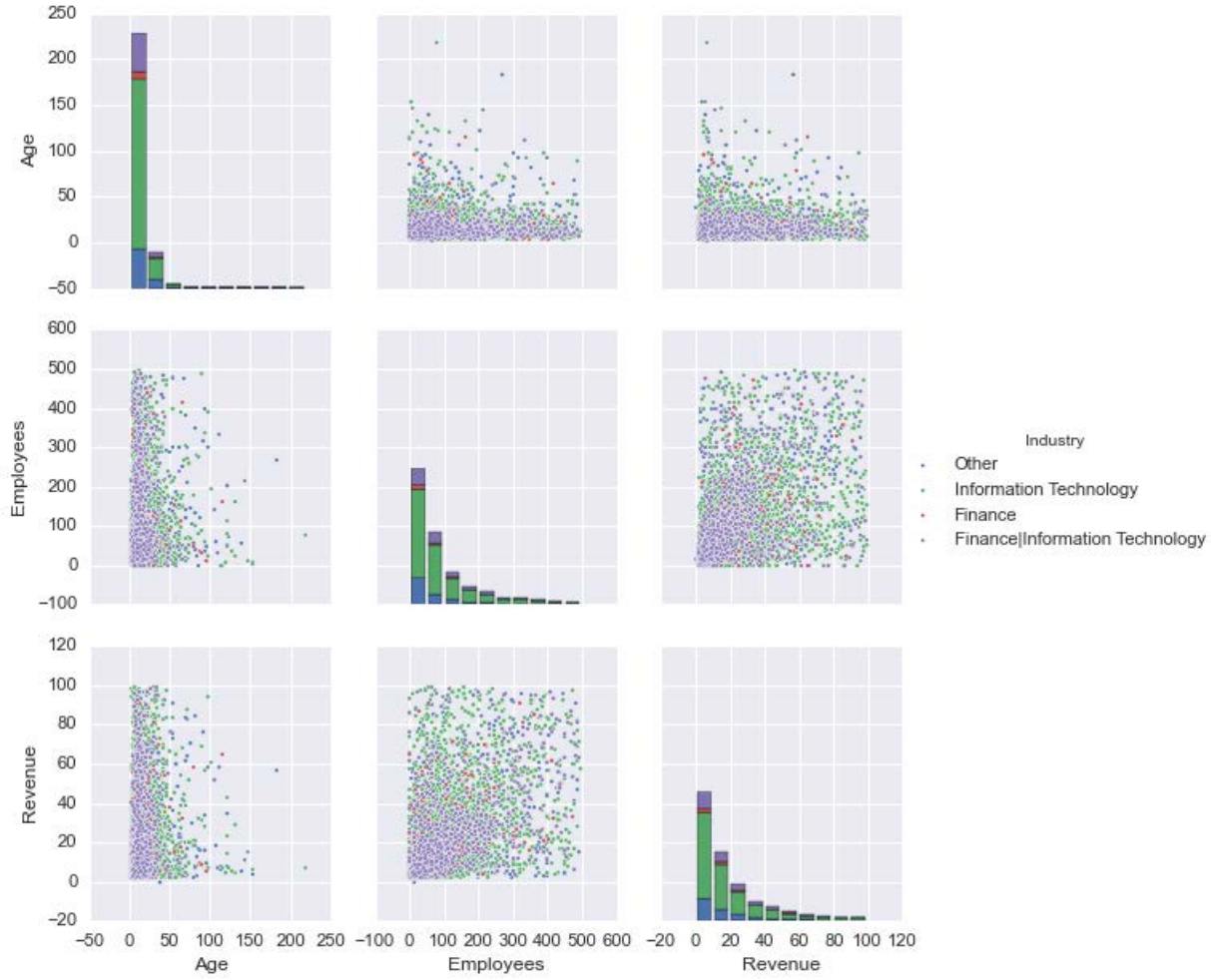


Figura A.5: Correlaciones entre ingresos, empleados y edad atendiendo distintas industrias

Apéndice B

Planificación

B.1. Tareas

En esta sección se expondrá la planificación de las tareas/actividades del proyecto. Concretamente, se va a mostrar la división de tareas realizada a priori así como el tiempo asignado a cada una de ellas.

En primer lugar, aunque el autor de esta memoria empezó sus prácticas en Global Incubator al acabar el verano de 2015, no empezó a trabajar en este proyecto concreto hasta Octubre ya que dedicó el tiempo a familiarizarse con la empresa. Así pues, a 15 de Octubre de 2015 se realizó la siguiente planificación: Se estimó que el proyecto duraría 24 semanas teniendo en cuenta los días festivos y vacaciones. Es decir, que para el 01 de Mayo ya estaría acabado. Estas fechas comprenden un total de 114 días laborables, de los cuáles 109 fueron efectivos. Por otro lado, de media se estimó gastar 4 horas a este trabajo (el resto se dedicaría a otros proyectos en la empresa) lo que en total harían un total de 436 horas dedicadas. Para simplificar, de ahora en adelante se utilizará como unidad temporal de medida una semana, teniendo en cuenta que eso equivale a 18,17 horas:

	Tiempo
Total proyecto 1 semana	24 semanas 18,7 horas
Total proyecto 1 día	109 días 4 horas
Total proyecto	436 horas
Total proyecto (real)	**455 horas**

Cuadro B.1: Longitud del proyecto

A continuación se describen las tareas en las que en un primer momento se desglosó el proyecto. Para que la estimación de las tareas así como la duración de las mismas fuese lo más precisa posible, el WBS (*Work breakdown structure*) siguiente no se formalizó hasta haber hecho un análisis preliminar de 4 días de duración acerca de los métodos de valoración (sección 2.1). No obstante,

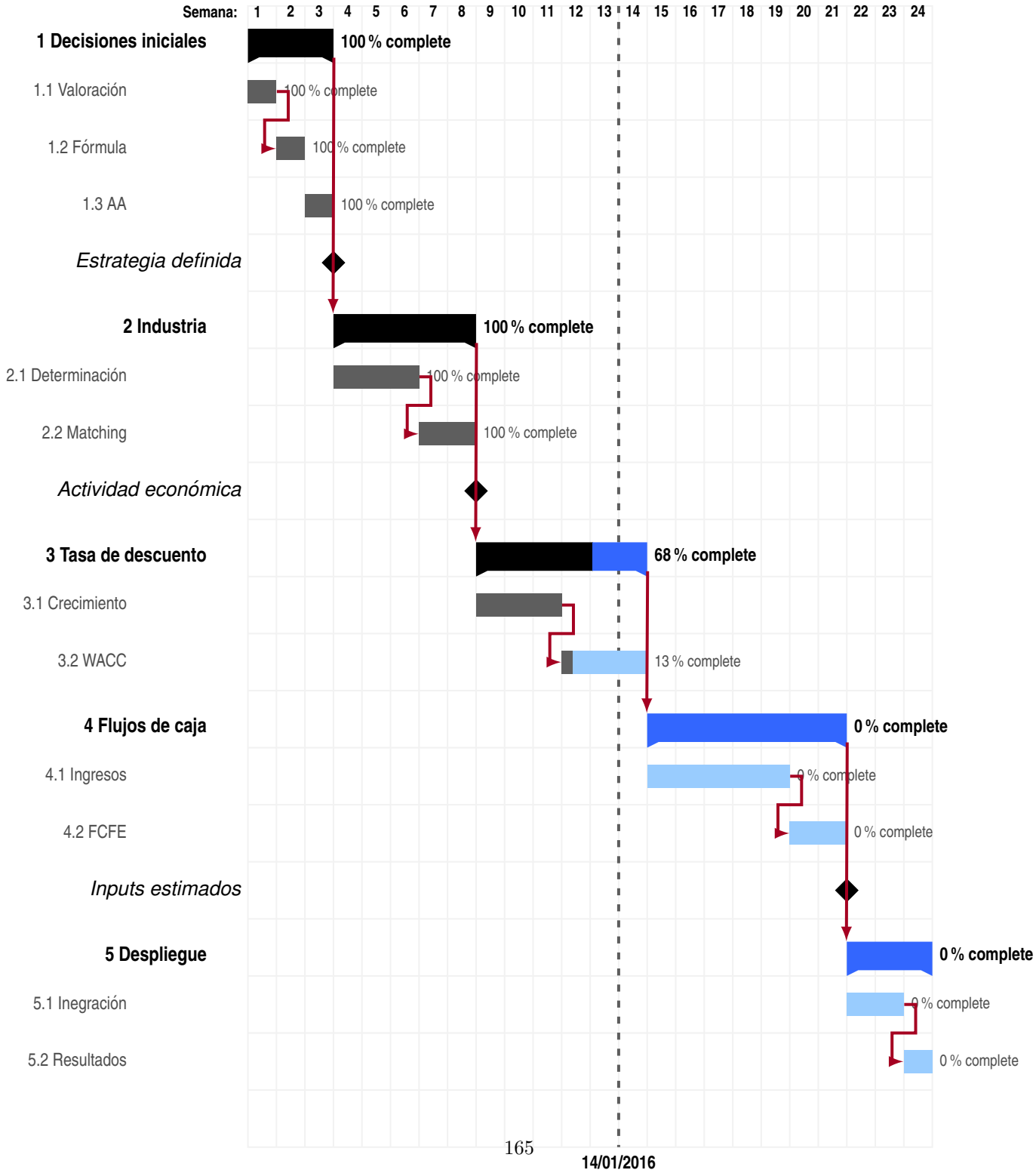
dicha tarea también se ha incluido en la planificación para poder visualizar el cuadro completo relativo al trabajo realizado:

1. **Decisiones Iniciales** (*3 semanas*)
   1.1. **Valoración** (*1 semana*): Se esclarecerán las bases que debe tomar el método de valoración a diseñar. Para ello se llevará a cabo un análisis de cuáles son los métodos de valoración que mejor resultados pueden ofrecer contextualizados bajo las peculiaridades del problema a resolver.
   1.2. **Fórmula** (*1 semana*): Se estudiarán y definirán los inputs de la fórmula de valoración.
   1.3. **AA** (*1 semana*): Se estudiará el estado del arte del Aprendizaje Automático ya que será necesario aplicarlo para conseguir algunos inputs.
2. **Industria** (*5 semanas*)
   2.1. **Determinación** (*3 semanas*): Se elaborará una funcionalidad del sistema capaz de determinar la actividad de una empresa (siguiendo una codificación predeterminada) a partir de un texto que la describa.
   2.2. **Matching** (*2 semanas*): Se elaborará un mecanismo capaz de unir datos de empresas de distintas fuentes que utilizan distintas codificaciones de sectores/industrias.
3. **Tasa de descuento** (*6 semanas*)
   3.1. **Crecimiento** (*3 semanas*): Se elaborará una funcionalidad del sistema capaz de estimar el crecimiento de una empresa a partir de los datos disponibles.
   3.2. **WACC** (*3 semanas*): Se elaborará una funcionalidad del sistema capaz de estimar el Coste Medio Ponderado del Capital (WACC) de una empresa a partir de los datos disponibles.
4. **Flujos de caja** (*7 semanas*)
   4.1. **Ingresos** (*5 semanas*): Se elaborará una funcionalidad del sistema capaz de estimar los ingresos de una empresa a partir de los datos disponibles. Se prevee que será necesario obtener datos externos para ello.
   4.2. **FCFE** (*2 semanas*): Se elaborará una funcionalidad del sistema capaz de deducir el importe necesario a los ingresos para que el valor final se ajuste lo máximo posible a los Flujos Libres de Caja.
5. **Despliegue** (*3 semanas*)
   5.1. **Integración** (*2 semanas*): Se unificarán todos los módulos desarrollados en las anteriores tareas para obtener una función única que realice todos los cálculos necesarios hasta resultar en una valoración.
   5.2. **Resultados** (*1 semanas*): Se aplicará el método de valoración implementado sobre un conjunto de empresas y se analizarán los resultados.

Además, se definen los siguientes hitos:

- **Estrategia definida** (3^a *semana*): En este punto se espera tener esclarecidos cuáles son los inputs de del método de valoración. Lo que permitirá empezar a trabajar en conseguir cada uno de ellos.
- **Actividad económica** (8^a *semana*): En este punto se será capaz de determinar cuál es la actividad de la empresa mediante una codificación entendida por el sistema. Es muy importante ya que todos los inputs de la fórmula dependen de ello de una manera u otra.
- **Inputs estimados** (21^a *semana*): En este punto el sistema debe de ser capaz de determinar por separado todos y cada uno de los inputs. Es importante porque una vez aquí lo más difícil habrá pasado y sólo quedará juntar las partes.

Ya se han asociado los distintos módulos del problema a un conjunto de tareas y se les ha asignado adecuadamente el recurso del tiempo. Por tanto, se mostrará en la siguiente página un Diagrama de Gantt (utilizando el paquete *pgfgantt* [Ska13] de LaTeX) donde, entre otras cosas, se aprecia cuáles son las relaciones de precedencia existentes entre las actividades. Se aprovecha dicho diagrama para ilustrar un retraso ocurrido en la actividad 2.1 experimentado en la semana 4 (explicado en la sección 5.1.5) y que fue un factor que motivó los 16 días de retraso del proyecto (se finalizó el 17 de Mayo). Teniendo en cuenta este contratiempo se contabilizan 455 horas de trabajo invertidas.


Semana: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
1 Decisiones iniciales
100 % complete
1.1 Valoración
100 % complete
1.2 Fórmula
100 % complete
1.3 AA
100 % complete
Estrategia definida
2 Industria
100 % complete
2.1 Determinación
100 % complete
2.2 Matching
100 % complete
Actividad económica
3 Tasa de descuento
68 % complete
3.1 Crecimiento
3.2 WACC
13 % complete
4 Flujos de caja
0 % complete
4.1 Ingresos
0 % complete
4.2 FCFE
0 % complete
Inputs estimados
5 Despliegue
0 % complete
5.1 Inegración
0 % complete
5.2 Resultados
0 % complete
14/01/2016

B.2. Presupuesto

A continuación se desglosarán los costes asociados al uso de los distintos tipos de recursos utilizados para llevar a cabo el proyecto (se toma como duración real del proyecto 6 meses). En la tabla B.2 se muestra el desglose de los costes del proyecto y en la B.3 el coste total con y sin IVA.

PERSONAL

Nombre	Categoría	Sueldo por hora	Horas trabajadas	Coste Total
Víctor García Cazorla	Investigador principal	20 €/h	455	9100 €
			Total:	9100 €

EQUIPAMENTO

Descripción	Coste	Uso dedicado al proyecto	Periodo de amortización	Coste Imputable
PC Portátil ASUS	623 €	64 %	48 meses	50 €
PC Portátil ASUS K52JU	530 €	36 %	48 meses	24 €
Monitor	154 €	64 %	84 meses	7 €
Servidor dedicado MG-256	323 €/mes	32 %	-	620 €
			Total:	701 €

SOFTWARE

Descripción	Coste	Uso dedicado al proyecto	Periodo de amortización	Coste Imputable
Licencia Windows 10	135 €	64 %	24 meses	22 €
Licencia Windows 8.1	99 €	36 %	22 meses	10 €
Microsoft Office 2010	250 €	10 %	36 meses	5 €
Software libre*	0 €	82 %	-	0 €
			Total:	37 €

Cuadro B.2: Desglose de los costes del proyecto

*Incluye: *Putty, Spyder, Notepad++, Debian OS*, LaTeX, *Selenium*

RESUMEN

Concepto	Coste
Personal	9100 €
Equipamento	701€
Software	37 €
Total sin IVA	9838 €
IVA (21 %)	2066 €
Total	11903 €

Cuadro B.3: Costes totales del proyecto